

# CAI-BASIC

A PROGRAM TO TEACH THE  
PROGRAMMING LANGUAGE 'BASIC'

THOMAS ANTHONY BARRY

LIBRARY  
NAVAL POSTGRADUATE SCHOOL  
MONTEREY, CALIF. 93940









# United States Naval Postgraduate School



## THE SIS

CAI-BASIC

A Program to Teach the  
Programming Language "BASIC"

by

Thomas Anthony Barry

Thesis Advisor

A.B. Roberts

September 1971

LIBRARY  
TULANE POSTGRADUATE SCHOOL  
NOVEMBER, CALIF. 93940



CAI-BASIC

A Program to Teach the  
Programming Language "BASIC"

by

Thomas Anthony Barry  
Lieutenant, United States Navy  
B. S. , United States Naval Academy, 1965

Submitted in partial fulfillment of the  
requirements for the degree of

MASTER OF SCIENCE IN COMPUTER SCIENCE

from the

NAVAL POSTGRADUATE SCHOOL  
September 1971



## ABSTRACT

This paper presents a computer aided instruction program that fulfills the objectives of teaching a simple programming language, interpreting student responses, and executing and editing student programs. The CAI-BASIC program is written in FORTRAN IV, level G, and executes on IBM- 2741 terminals while running under the CP-67/CMS time sharing system on the U.S. Naval Postgraduate School's IBM-360/67 computer system. The instructional phase of CAI-BASIC presents the fundamentals of "BASIC," a simple user oriented language, in seven lessons. During the instructional sessions the student is presented material and, based on his response to questions, he is routed to the next sequence of instructions. The execution phase of CAI-BASIC allows execution of "BASIC" programs, and has an optional debug feature that provides a trace of program variables to aid the student in finding programming errors. In the event of programming errors the user may enter an edit mode to correct mistakes in his program.



## TABLE OF CONTENTS

|      |                                                  |     |
|------|--------------------------------------------------|-----|
| I.   | INTRODUCTION .....                               | 4   |
|      | A. DEFINITION OF CAI.....                        | 4   |
|      | B. CAI DEVELOPMENTS .....                        | 4   |
|      | C. OBJECTIVES .....                              | 6   |
| II.  | DESCRIPTION OF CAI-BASIC .....                   | 7   |
|      | A. INSTRUCTION PHASE .....                       | 7   |
|      | B. EXECUTION PHASE.....                          | 8   |
| III. | LIMITATIONS AND EXTENSIONS .....                 | 10  |
| IV.  | CONCLUSION .....                                 | 12  |
|      | APPENDIX A USING CAI-BASIC .....                 | 13  |
|      | APPENDIX B CAI-BASIC PROGRAM.....                | 16  |
|      | APPENDIX C TERMINAL SESSION WITH CAI-BASIC ..... | 70  |
|      | BIBLIOGRAPHY.....                                | 119 |
|      | INITIAL DISTRIBUTION LIST .....                  | 120 |
|      | FORM DD 1473 .....                               | 121 |



## I. INTRODUCTION

During the last few years many sophisticated projects in Computer Assisted Instruction (CAI) have evolved. The potential of this area has captured the imagination of researchers and the public at large. Yet the U. S. COMMISSION ON INSTRUCTIONAL TECHNOLOGY (ref. 9) found that the status of instructional technology in American Education was low in both quantity and quality. There are many reasons that CAI has not been accepted with any marked degree of enthusiasm by educators in general. The major reasons are the exorbitant costs of individualized instruction and illusions over just what CAI is.

### A. DEFINITION OF CAI

Computer Assisted Instruction is machine augmented instruction that differs from Programmed Instruction or simple page turning machines in that the computer system has logic and memory capabilities to assist in the instructional process. The main advantage in CAI is that it can individualize education. A student can proceed at his own pace, spending more time on difficult material, and quickly covering material which comes easy to him.

### B. CAI DEVELOPMENTS

A. G. OETTINGER, in his book RUN, COMPUTER, RUN (ref. 6) states that educators and the computer industry are equally to blame for the failure of CAI to realize its potential. Instructional technology has been force fed,





oversold, and prematurely applied; and as a result the educators are wary of false promises. OETTINGER feels that colleges and universities will be an effective proving ground for future educational technology. However, at the present time, of the thousands of colleges and universities in the nation only twenty-five are considered major CAI centers.<sup>1</sup>

A common problem confronting the nation's colleges and universities is that of providing instruction in elementary computer programming.<sup>2</sup> The popularity of the computer science field, and the range of application of computers in everyday life means that the demand for people who know how to communicate with computers is growing rapidly. The solution to the problem is the computer itself. "Whatever the state of CAI with respect to other subjects, the computer is the ideal instrument for teaching its own use."<sup>3</sup>

W. R. SMITH and J. L. YOUNG, graduate students at the Naval Postgraduate School, presented a proposal for the use of computer Assisted Instruction at the Naval Postgraduate School (ref.10). As this introduction has done, they created an awareness for the need of CAI, and they made specific recommendations to provide a balanced program for the entire community to emulate. In particular they recommended commencing student projects to probe the potentials of CAI

---

1. STOLUROW, L. M. , "Computer Assisted Instruction," REPORT OF A CONFERENCE, U. S. Department of HEW, p. 49, 1969.

2. FENICHEL, R. R. , WEIZENBAUM, J. , AND YOCHELSON, J. C. "Program to Teach Programming," COMMUNICATION OF THE ACM, v.13, p.141, March 1970.

3. IBID. , p.141-142.



at the Naval Postgraduate School.

It was with these ideas about using the computer to teach computer programming and providing a CAI base at the Naval Postgraduate School that CAI-BASIC was undertaken.

### C. OBJECTIVES

The objective of CAI-BASIC was to develop a CAI program to teach a computer programming language on the Naval Postgraduate School's time-sharing terminals. The programming language chosen was "BASIC" (Beginners All-purpose Symbolic Instruction Code), a computer language developed at Dartmouth College in the mid-1960's. "BASIC" was selected because of its simplicity yet fairly large range of application for the user who wants to take advantage of computer processing. It was felt that "BASIC" would provide the average non-computer oriented graduate student with satisfactory results in a minimal amount of study time.

With this objective in mind three sub-goals were determined to be essential for the CAI-BASIC project: to be able to interpret student responses, to be able to execute "BASIC" programs in order to give the student sufficient programming practice, and to be able to allow editing of "BASIC" programs.



## II. DESCRIPTION OF CAI-BASIC

The CAI-BASIC program is written in FORTRAN IV, level G, and executes on IBM-2741 terminals while running under CP-67/CMS Time Sharing System on the Naval Postgraduate School's IBM-360/67 Computer System. The "BASIC" language implemented in CAI-BASIC is standard non-extended "BASIC" (refs. 3, 5, 10).

The CAI-BASIC program consists of two phases; an instruction phase, and an execution and editing phase. A detailed discussion of the use of CAI-BASIC is presented in APPENDIX A.

### A. INSTRUCTION PHASE

The instruction phase presents fundamental concepts common to most programming languages: identifiers, variables, iteration, branching, sub-routines, built in functions, and recursion. The instruction set consists of seven lesson modules which, for reasons of coherency, are dependent upon each other. The underlying philosophy in preparing the lesson modules was that an important idea should be presented only after a need for it has clearly been established.<sup>4</sup>

Each lesson consists of instruction sequences followed by questions, evaluation of student responses, routing to the next sequence of instruction,

---

4. FENICHEL, OP. CIT., p. 142.



and a summary of the lesson followed by questions and/or problems to program. The student is allowed to progress through the lessons at his own speed, and allowed to review lessons or execute programs at the completion of each lesson.

Student responses are interpreted by one of two methods depending upon the type of question asked. When the question is of the true-false, multiple choice or give the answer type then the response is compared with a pre-stored result. When the student is asked to reply with a "BASIC" statement or to write a "BASIC" program, the CAI-BASIC compiler interprets the "BASIC" statements for syntactic correctness. When the student writes a program he can compare his answer with a pre-stored result, and if the answer is wrong the student is shown a solution program.

## B. EXECUTION PHASE

The heart of the execution phase is the "BASIC" compiler that is used at the Naval Postgraduate School's Computer Center to execute "BASIC" programs in a batch mode. This compiler, written in FORTRAN IV, level G, was modified from a batch processing compiler to a line-by-line interpreter and incorporated into the CAI-BASIC program, and called the CAI-BASIC COMPILER.

The CAI-BASIC COMPILER has an added feature to aid students in debugging programs on the terminal. The added feature is called a "DEBUG" function and it produces a list of numeric and alpha-numeric data that were put into the program; and, in addition it produces a trace of all simple variables as they are assigned values during program execution.





At the completion of program execution or when an execution error occurs, the student has an option to enter the CAI-BASIC Edit mode to correct his program one line at a time. When all editing is completed the program is executed again.



### III. LIMITATIONS AND EXTENSIONS

The major limitation of CAI-BASIC is the absence of any supervision over the student's progress as he proceeds through the instruction phase. A possible extension to this project would be to add a supervisory routine to maintain the student's progress and to keep a record of his errors. Thus, having the student's progress level and error record, CAI-BASIC could be tailored to instruct the student at his own learning level. In other words, CAI-BASIC could be made into a more completely interactive teaching program (refs. 1, 4).

An additional feature that would aid the student interaction is a communication link between the student and a professor so that student questions can be answered. R. R. FENICHEL (refs. 1, 2) described how students enter a special mode to type questions during the instruction session on the terminal. Then at the beginning of the student's next instruction session, all of his previous questions are answered on the terminal. FENICHEL refers to this as the "mailbox" system. Students type questions for the "mailbox" and the professor replies with answers or pertinent information for the "mailbox."

Another limitation to the existing CAI-BASIC program is the inability of CAI-BASIC to monitor the student when he enters the execution phase to write programs. The CAI-BASIC system does not oversee the student, beyond checking for syntax errors in his program. There is no method of inspecting his



programs from a tutorial point of view in order to help with semantic programming errors.

A foreseeable development to bring CAI-BASIC into the Artificial Intelligence field would be to make CAI-BASIC an "intelligent" tutor. This would remove the present inflexibility of interpreting student responses with pre-stored answers and open the possibility of CAI-BASIC understanding the logic of student responses and student programs.



#### IV. CONCLUSION

As stated in the objectives, the goal of CAI-BASIC was to develop a CAI program to teach a computer programming language on the Naval Post-graduate School's terminal system. The technical aim of completing the project and bringing it to an operating level has been met. However, an evaluation of the practicality of the project is still in the speculative stage.

Several computer and non-computer oriented students have tried CAI-BASIC and they found it to be both understandable and beneficial, but a full scale evaluation of the effectiveness of CAI-BASIC as a teaching tool is not possible from this limited sampling. It is sufficient to say that initial indications are that CAI-BASIC can provide a satisfactory means of learning a simple but capable programming language in a minimal amount of time. The student's participation in the learning experience encourages him to teach himself and to practice new programming skills. It is hoped that future use of CAI-BASIC by students will demonstrate its practicality.

Although the scope of CAI-BASIC was not particularly broad, the results obtained provide a basis for future CAI projects. The foundation has been laid, and the limitations and extensions of the previous section suggest a direction for future CAI efforts. Many previous CAI projects have suffered from too many or too large a scope of objectives, and as a result they never realize any practical results. This project has met its objectives and, hopefully, future CAI projects will benefit from its results.





## APPENDIX A

In its present configuration the CAI-BASIC program occupies approximately ten cylinders of private disk space and requires an overlaying routine to execute under the control of CP-67/CMS which limits users to a 175K virtual machine. The overlaying routine, written by a Naval Postgraduate School Computer Center system programmer, initially loads only lesson 1 when CAI-BASIC is loaded into core. When any lesson other than lesson 1 is used, it is overlaid onto lesson 1.

The overlaying routine allows students to log on to CP-67/CMS as general users and to link into CAI-BASIC. This allows CAI-BASIC to be available to all of the terminal stations. The user gains access to CAI-BASIC by logging on the terminal as follows:

```
login yyyygxx      (xx is the terminal nr.)
                   (yyyy is your user nr.)
ENTER PASSWORD
npg
ENTER 4-DIGIT PROJECT NUMBER.... ETC.
0623cs04
READY AT (TIME) ON (DATE)
CP
link 0909p 191 192
ENTER PASSWORD
teach
SET TO READ ONLY
i cms
CMS.. VERSION 01/21/71
login 192 t,p
**T (192) READ ONLY**
teach
```



Small print is typed in by the user, and the capitalized print is the computer terminal response.

The CAI-BASIC program consists of two phases, an instruction phase and an execution and editing phase. After logging on to the system, the user enters the main routine, CAIBAS, which directs the general flow of CAI-BASIC.

If the user has not used CAI-BASIC before he enters the instruction phase, he is presented with an introduction to CAI-BASIC, and then is sequentially guided through the lessons. After each lesson the user is given the opportunity to terminate his session, to review a lesson, to enter the execution phase, or to go on to the next lesson.

If the user has used CAI-BASIC before, he is given the opportunity to review any lesson or to go to the next lesson in his learning sequence.

When the execution and editing routine, TEST1, is entered the user is allowed to execute standard "BASIC" programs (refs. 3,5,10). The Test1 routine incorporates the "BASIC" compiler, COMPLR, which interprets each "BASIC" input statement. Each "BASIC" statement is input one line at a time; and, if a syntax error occurs on the input, COMPLR prints an error message. The user can then input the correct "BASIC" statement. When the "end" statement is input, the program is checked for global errors; and, if none occur, the program is executed.

When global errors or execution errors occur, the user is given the choice of entering the edit mode of TEST1 to correct his program, or using the "DEBUG" feature to find his programming errors.

The edit mode allows the user to delete, add to, or correct "BASIC"



statements in his program one line at a time. The user is given a listing of his program with reference numbers for editing purposes. To edit a program the user is first asked for the statement reference number, and then asked for the specific edit command.

A statement is deleted by typing in the reference number of the statement to be deleted, hitting the carriage return, and then typing in the command "DEL."

A statement is added 'after' the reference number typed in by typing in the command "ADD1" followed by the "BASIC" statement to be added. To add a statement before the first statement in the program, the reference number zero (0) is used.

A statement is corrected by typing in the reference number of the statement to be corrected, hitting carriage return, and then typing in the 'entire' correct "BASIC" statement.

The "DEBUG" feature is designed to provide useful information to the user who has encountered an execution error in his program. It is used by adding the key word "DEBUG" as a statement in the user's program.

The "DEBUG" feature gives the user a list of all numeric and alpha-numeric data that were used in the program; and, in addition, it produces a trace of all simple variables with their values as they are assigned values during program execution.



# CAI-BASIC : A PROGRAM TO TEACH THE PROGRAMMING LANGUAGE "BASIC"

T.A. BARRY

U.S.NPS. 8/17/71

THE FILES USED IN CAI-BASIC UNDER CP-67/CMS ARE AS FOLLOWS :

MAIN PROGRAM----- CAIBAS

SUBROUTINES-----INITIAL  
CRUNCH  
TEST1  
LESON1--> LESC7

OVERLAYING ROUTINES FOR CP-67/CMS :

SEARCH ALOAD EXIT

THE FOLLOWING SUBROUTINES ARE PART OF THE CAIBASIC COMPILER,  
AS MODIFIED FROM THE NPS "BASIC" COMPILER USED FOR BATCH PROGRAMS,  
BUT ARE NOT INCLUDED AS PART OF THE CAI-BASIC PROGRAM :

|        |        |       |       |
|--------|--------|-------|-------|
| ACON   | COMPLR | ERR   | INSNO |
| APRINT | CUNV   | DUMMY | LTL   |
| BUFFIL | CPRINT | EVAL  | RANF  |
| CEND   | DIM    | EXEC  | TEST  |

SYMBOL TABLE : (GLOBAL SYMBOLS AND VARIABLES)

|           |        |                 |         |         |        |        |                |
|-----------|--------|-----------------|---------|---------|--------|--------|----------------|
| CARD      | VECTOR | HOLDING         | STUDENT | INPUT   | WITH   | BLANKS | REMOVED        |
| CARDP     | VECTOR | HOLDING         | STUDENT | INPUT   |        |        |                |
| ILNGTH    | LENGTH | OF              | INPUT   | IN      | VECTOR | CARDP  |                |
| ASTRSK    | WHEN   | CARDP(1)=ASTRSK | THE     | USER    | HAS    | MADE   | A TYPING ERROR |
| ALPHA     | VECTOR | HOLDING         | THE     | LETTERS | OF     | THE    | ALPHABET       |
| DIGIT     | VECTOR | HOLDING         | THE     | DIGITS  | 0-->9  |        |                |
| ALPHA(25) | THE    | LETTER          | 'Y'     |         |        |        |                |
| ALPHA(14) | THE    | LETTER          | 'N'     |         |        |        |                |





CCCCCCCCCCCCCCCCCCCC

CAI000010  
CAI000030  
CAI000040  
CAI000050  
CAI000060

CAIBAS IS THE MAIN PROGRAM. IT IS A COMPUTER AIDED INSTRUCTION  
PROGRAM WHICH TEACHES FUNDAMENTAL OF THE PROGRAMMING CONCEPTS USING  
THE LANGUAGE BASIC. THE HEART OF THE PROGRAM PROVIDES INSTRUCTION  
AND INTERPRETES STUDENT RESPONSES. SOME ACDED FEATURES ARE  
EXECUTION AND EDITING OF BASIC PROGRAMS IN "TEST1"  
SUBROUTINES CALLED FROM CAIBAS ARE:

INITIAL  
CRUNCH  
TEST1  
LESON1 --> LESON7  
SEARCH  
ALOAD

SYMBOL TABLE : (LOCAL SYMBOLS AND VARIABLES)

LSNUM CURRENT LESSON NUMBER  
OLD FLAG. SET TO 1 IF USER HAS USED CAI-BASIC BEFORE  
LES1 --> LES7 VARIABLES FOR OVERLAYING ROUTINES

CAI000070  
CAI000080  
CAI000090  
CAI000100  
CAI000110  
CAI000120  
CAI000130  
CAI000140  
CAI000150

COMMON  
STACK(100), PROG(2000), CARD(80), CARDP(80), ALPHA(48),  
IAPTR, INPTR, IADATA(500), XNDATA(500), STRING(5),  
DIGIT(10), IPRTB(10), LISTST(100), ISTLST(100),  
PRT(2500), NERRS, INST, NSTLST, DEBUG, DOLSGN, QUOTE,  
EQUALS, PARRT, DECIMAL, PLUS, CMINUS, SLASH, COMMA,  
PARLET, ASTRSK, BLANK  
INTERP, EXERR

CMCMGN  
REAL\*8 LES1, LESON1, //  
REAL\*8 LES2, LESON2, //  
REAL\*8 LES3, LESON3, //  
REAL\*8 LES4, LESON4, //  
REAL\*8 LES5, LESON5, //  
REAL\*8 LES6, LESON6, //  
REAL\*8 LES7, LESON7, //  
CALL SEARCH  
CALL INITIAL  
LSNUM=0  
OLD=0

CAI000160

CAI000180  
CAI000190  
CAI000200  
CAI000210  
CAI000230  
CAI000240

99  
WRITE(6,99), HI, WELCOME TO CAI-BASIC. THERE ARE ONLY A FEW  
FORMAT(0,0), HI, REMEMBER IN CRDOR TO HAVE A SUCCESSFUL SESSION.  
SIMPLE RULES TO REMEMBER IN CRDOR : TYPE IN THE CORRECT REPLY AND  
ON THE TERMINAL WITH CAI-BASIC : TYPE IN THE CORRECT REPLY AND  
1. WHEN ASKED FOR RETURN KEY ON THE RIGHT SIDE OF THE KEYBOARD.  
HIT THE CARRIAGE RETURN KEY WHILE MAKING ANY RESPONSE  
\*\* 5X, 2. IF YOU MAKE A TYPING ERROR WHILE MAKING ANY RESPONSE



```

*OR INPUT , TYPE IN FOUR DOLLAR SIGNS ($$$$ ) AFTER THE ERROR OR'// CAI00260
*ANYWHERE ON THAT INPUT LINE AND HIT CARRIAGE RETURN .THE CAI00270
*ENTIRE LINE WILL THEN BE IGNORED AND YOU CAN TYPE IN THE CORRECT'// CAI00280
*1 INPUT OR RESPONSE . IN THE SESSION YOU WANT TO STOP THE SESSION'// 5X,
*3. IF AT ANY POINT .QUIT. AS SOON AS YOU ARE ASKED FOR THE NEXT'// CAI00300
*TYPE IN THE WORD .QUIT. THEN HIT ATTN KEY AND TYPE LOGOUT CAI00310
*RESPONSE , HIT CARRIAGE RETURN, THEN HIT ATTN KEY AND TYPE LOGOUT
*4. DURING YOUR TERMINAL SESSION CAI-BASIC WILL HALT OCCASIONALLY'// 5X,
*TO LET YOU READ A SEQUENCE OF INFORMATION . WHEN YOU ARE'// CAI00330
*READY TO CONTINUE , TYPE IN 'GO. ' , AND HIT CARRIAGE RETURN'// 5X CAI00340
*5. DURING YOUR TERMINAL SESSION YOU MAY NOTICE THAT'// CAI00360
*THE TYPING IS NOT ALWAYS PERFECT . SOME DAYS THE COMPUTER IS'// CAI00370
*NOT UP TO PAR AND YOU WILL HAVE TO ADJUST TO THE MINOR IRRITANT'//
*IF YOU ONLY WANT TO EXECUTE PROGRAMS AT THIS TIME THEN
*REPLY : YES ; OTHERWISE REPLY : NO . '///)
*READ(5,101,END=300)CARD
*CALL CRUNCH(1,LENGTH)
*IF(CARDP(1).EQ.ASTRSK) GO TO 1
*IF(CARDP(1).NE.ALPHA(25)) GO TO 2
EXECUTION ONLY PHASE
CALL TEST1
WRITE(6,109)
READ(5,101,END=401)CARD
CALL CRUNCH(1,LENGTH)
*IF(CARDP(1).EQ.ALPHA(25)) RETURN
*IF (CARDP(1).EQ.ALPHA(14)) GO TO 402
*WRITE(6,104)
*GO TO 400
*IF(CARDP(1).EQ.ALPHA(14)) GO TO 3
*WRITE(6,104)
*GO TO 1
*WRITE(6,100)
*FORMAT(5,101,END=302)CARD
*REPLY: YES ; OTHERWISE REPLY: NO . '///)
*4 READ(5,101,END=302)CARD
*FORMAT(80A1)
*CALL CRUNCH(1,LENGTH)
*IF(CARDP(1).NE.ALPHA(14))GO TO 20
OLD USER. DETERMINE PROGRESS LEVEL AND ROUTE TO DESIRED LESSON
402 OLD=1
102 WRITE(6,102);
FORMAT(6,102);
NOW YOU MAY CHOOSE TO REVIEW ANY LESSONS'//

```



```

*OR TO BEGIN WHERE YOU ENDED DURING YOUR LAST SESSION.**)
WRITE(6,105)
FORMAT(0,'10X,'THE LESSONS ARE AS FOLLOWS : '//10X,' LESSON 1
105 *PROGRAM FORMAT AND BASIC DEFINITIONS$
* AND DATA//10X,' LESSON 3 ASSIGNMENT STATEMENTS AND BUILT IN FUNCT
* TIONS//10X,' LESSON 4 BRANCHING //11X,'LESSON 5 LOOPING AND SUBSCR
* TPTED VARIABLES//10X,' LESSON 6 SUBROUTINES AND RECURSION //10X,'
* LESSON 7 SUMMARY OF BASIC STATEMENTS
* //*** WHEN YOU ARE READY TO CONTINUE , TYPE IN GO AND HIT //*
* THE CARRIAGE RETURN ***/)
READ(5,101,END=5)CARD
101 WRITE(6,103)
103 FORMAT(0,' 5X,' REPLY WITH THE NUMBER OF THE LESSON YOU WISH TO C
* OVER//)
6 READ(5,101,END=301)CARD
CALL CRUNCH(ILNGTH)
DO 7 I=2,8
IF (CARDP(1).EQ.DIGIT(I)) GO TO 9
7 CONTINUE
301 WRITE(6,104)
104 *FORMAT(0,'*** YOUR REPLY WAS TYPED INCORRECTLY; CHECK THE '//
* QUESTION AND REPLY AGAIN ***/)
GO TO 6
9 LSNUM=I-1
10 GC TO (11,12,13,14,15,16,17),LSNUM
11 CALL LSONI
12 GO TO 25
12 CALL ALOAD(LES2,N1)
13 GO TO 25
13 CALL ALOAD(LES3,N1)
14 GO TO 25
14 CALL ALOAD(LES4,N1)
15 GO TO 25
15 CALL ALOAD(LES5,N1)
16 GO TO 25
16 CALL ALOAD(LES6,N1)
17 GO TO 25
17 CALL ALOAD(LES7,N1)
GO TO 25
INSTRUCTION PHASE ** NEW USER
IF (CARDP(1).EQ.ALPHA(25))GO TO 21
302 WRITE(6,104)
GO TO 4
21 WRITE(6,106)
106 FORMAT(0,'10X,'CAIBASIC IS A PROGRAM TO TEACH YOU THE
CAI00660
CAI00670
CAI00680
CAI00700
CAI00730
CAI00740
CAI00750
CAI00760
CAI00780
CAI00790
CAI00800
CAI00820
CAI00830
CAI00840
CAI00850
CAI00860
CAI00870
CAI00900
CAI00910
CAI00930
CAI00950
CAI00970
CAI00990
CAI01010
CAI01030
CAI01040
CAI01060
CAI01070
CAI01080
CAI01090
CAI01100
CAI01110

```

۷۷۷





```

*FUNDAMENTALS OF A PROGRAMMING LANGUAGE. THE LANGUAGE TO BE LEARNED. CAI01120
*// IS BASIC ; A SIMPLE LANGUAGE FOR THE USER WHO HAS LITTLE.// KNOW CAI01130
*WLEDGE OF COMPUTERS AND WHOSE PRIMARY INTEREST IS IN OBTAINING.//
*RESULTS.//1CX, THE SIMPLICITY OF THE BASIC LANGUAGE AND ITS RANGE. CAI01150
*// OF CAPABILITIES SHOULD ALLOW YOU TO LEARN THE LANGUAGE AND.// CAI01160
*WRITE RECOMMENDED FOR A MINIMAL AMOUNT OF TIME.//5X, THE REFERENCE T
*EXTS IN # 0211-12 APRIL 1971.// ( FREE UPON REQUEST IN I-147 )//75X
*// 2. INTRODUCTION TO COMPUTING LIBRARY.//5X, 1. BASIC LANGUAGE, R.L.
*// NOLAN.// (BOOKSTORE / MAIN LIBRARY)//5X, 3. BASIC PROGRAMMING,
*// V.C. HARE.// (COMPUTER CENTER LIBRARY) //***** AFTER YOU HAVE CAI01170
*// FINISHED READING AN INPUT, TYPE IN GO AND HIT THE RETURN KEY.// CAI01180
*// AND THE PROGRAM WILL CONTINUE ***.// CAI01190
*// READ(5,101,END=307)CARD CAI01200
307 WRITE(6,107) CAI01210
107 FORMAT(0,1CX, DURING YOUR TERMINAL SESSION YOU WILL BE LEARNING.// CAI01220
*// THE STRUCTURE OF THE LANGUAGE BASIC. THE INSTRUCTION SET.// CAI01230
*CONTAINS 7 LESSONS AND YOU MAY PROCEED THROUGH THE LESSONS AT.// CAI01240
*YOUR OWN SPEED.// CAI01250
*WRITE(6,105) CAI01260
308 READ(5,101,END=308)CARD CAI01270
108 WRITE(6,108) CAI01280
*// IN EACH LESSON YOU WILL BE GIVEN INSTRUCTION.// CAI01290
*SEQUENCES AND THEN YOU WILL BE ASKED QUESTIONS TO SEE IF YOU.// CAI01300
*UNDERSTOOD THE INSTRUCTIONS. THE QUESTIONS WILL BE OF VARIOUS.// CAI01310
*TYPES: MULTIPLE CHOICE, TRUE/FALSE, ACTUAL PROGRAM STATEMENTS, ETC.// CAI01320
*// YOU WILL BE PROMPTED FOR YOUR ANSWER, AND WHEN READY TYPE IN.//
*YOUR RESPONSE AND HIT THE RETURN KEY.//10X, IF YOU KEEP THE TELETYPE CAI01340
*// YOUR OUTPUT FROM FUTURE USE.// CAI01350
*// READ(5,101,END=309)CARD CAI01360
309 LSNUM=1 CAI01370
24 IF (LSNUM.GT.7) GO TO 40 CAI01380
25 GO TO (11,12,13,14,15,16,17),LSNUM
109 WRITE(6,109)
*// DO YOU WANT TO TERMINATE YOUR INSTRUCTION SESSION CAI01410
26 ?// (REPLY: YES OR NO) CAI01420
*// (REPLY: YES OR NO) CAI01430
26 READ(5,101,END=310)CARD CAI01440
*// CRUNCH(LENGTH) CAI01450
CALL CRUNCH(1).EQ.ALPHA(25)) GO TO 40 CAI01460
IF (CARDP(1).EQ.ALPHA(14)) GO TO 30 CAI01470
310 WRITE(6,104) CAI01480
26 GO TO 26 CAI01490
C OLD USERS PICK NEXT LESSON ; NEW USERS GET NEXT LESSON IN SEQUENCE CAI01500
30 WRITE(6,209) CAI01510
209 FORMAT(0,1CX, IF YOU WANT TO EXECUTE PROGRAMS AT THIS TIME.// CAI01520
*// THEN REPLY: YES ; OTHERWISE REPLY: NO AND THE INSTRUCTION.// CAI01540
*WILL CONTINUE.//)

```









CCCCCCCC

```
KTK
IARRAY
DEBUG
OLD
INBIG
NFOR

COUNTER TO DETERMINE IF REFERENCE NUMBER IS 1 OR 2
DIGITS
VARIABLE USED IN CAI-BASIC COMPILER
FLAG " SET TO 1 WHEN USER HAS EDITED FIRST PROGRAM
" " " " " "
```

```
COMMON
DIMENSION SFILE(100,80)

STACK(100), PROG(2000), CARD(80), CARDP(80), ALPHA(48),
IAPTR, INPTR, IADATA(500), XNDATA(500), STRING(5),
DIGIT(10), IPRIB(10), LISTST(100), IISTST(100),
PRT(2500), NERRS, INST, NSTLST, DEBUG, DOLSGN, QUOTE,
EQUALS, PARRT, DECIMAL, PLUS, CMINUS, SLASH, COMMA,
PARLET, ASTRSK, BLANK
COMMON INTERP, IEXERR
```

CCCCCCCC

EXECUTION PHASE

IF NEW SET DON'T PRINT INTRO AGAIN

IF(NEW.EQ.1)GO TO 3

99

```
WRITE(6,99)
FORMAT(0,15X, ' ***** CAIBASIC COMPILER ***** THE BASIC '
* THE CAIBASIC ACCEPTS STANDARD BASIC BY LINE INTERPRETER , AND IT ANALYZES '
* EACH BASIC STATEMENT AS IT IS INPUT AND AN ADDED FEATURE OF '
* THAT ALLOWS THE USER TO ADD , DELETE AND CORRECT STATEMENTS ; '
* AND TO GET A LISTING OF ALPHA-NUMERIC AND NUMERIC DATA USED , '
* AND A TRACE OF ALL SIMPLE VARIABLES AS THEY ARE ASSIGNED VALUES , '
* IN THE PROGRAM. IS USED BY ADDING THE KEY WORD DEBUG , '
* AS A STATEMENT FEAT TO YOUR PROGRAM. '
* THE EDITING MODE IS AVAILABLE TO THE USER WHEN AN EXECUTION , '
* ERROR OCCURS AND AFTER SUCCESSFUL EXECUTION , '
* IN THE EVENT OF AN INPUT ERROR THE INTERPRETER WILL ANALYZE '
* THE ERROR AND PRINT AN ERROR MESSAGE. IF THIS OCCURS FIND THE '
* ERROR OF INPUT THE CORRECT BASIC STATEMENT FOR THE CURRENT '
* LINE (NOTE : TYPING ERRORS CAN BE DELETED BY TYPING FOUR DOLLAR '
* SIGNS ($$$$) ON THE SAME LINE AS THE ERROR , HITTING CARRIAGE '
* RETURN , AND INPUTTING THE LINE AGAIN .) '
OLD=0
309 WRITE(6,300)
FORMAT(0,10X, ' ***** CAIBASIC EXECUTION ***** ')
```

309



```

CALL INITIAL
NEW=1
DEBUG=C.0
NSTMT=0
IEND=0
INBIG=1
IARRAY=808
NFOR=0
INTERP=0
IEXERR=0
5 WRITE(6,100)
100 FORMAT(0,5X,' INPUT BASIC PROGRAM NOW (ONE LINE AT A TIME )'///)
10 READ(5,101,END=5)CARD
101 FORMAT(80A1)
CALL CRUNCH(ILNGTH)
IF(CARDP(1).EQ.ASTRSK) GO TO 10
FIND ' END ' STATEMENT
C
C
C
699 DC 699 I=1,80
700 IF(CARDP(1)).NE.ALPHA(5).OR.CARDP(I+1).NE.ALPHA(14))GO TO 699
IF(CARDP(I+2).EQ.ALPHA(4).AND.CARDP(I+3).EQ.BLANK)GO TO 701
CCCONTINUE
CALL CCMPLR(NFOR,IARRAY,ILNGTH,INBIG)
C
C
C
IF NO ERRORS , FILE INPUT STATEMENT
IF(NERRS.EQ.0) GO TO 9
IF(IEXERR.EQ.1)GO TO 9
NERRS=0
GO TO 10
C
C
C
PUT IN END STATEMENT
701 NSTMT=NSTMT+1
IEND=1
DO 702 I=1,80
702 SFILF(NSTMT,I)=CARD(I)
GO TO 700
9 IF(NSTMT.GT.99) GO TO 511
IF(IEND.EQ.1) GO TO 12
NSTMT=NSTMT+1
DO 11 J=1,80
11 SFILF(NSTMT,J)=CARD(J)
GO TO 10
511 WRITE(6,512)
512 FORMAT(0,' YOU HAVE REACHED THE MAXIMUM PROGRAM SIZE ALLOWED.'/
*UNDER CAI-BASIC . YOU WILL HAVE TO MODIFY YOUR PROGRAM TO

```



```

C
C
C
*STAY BELOW 100 PROGRAM STATEMENTS .'/')
GO TO 513
EXECUTION ERROR GO TO EDIT ROUTINE
12 IF(IEXERR.EQ.1)GO TO 31
153 WRITE(6,205)
205 FORMAT('O',I0X,' DO YOU WANT TO EDIT YOUR PROGRAM '/, REPLY : YE
*S OR NO .')
312 READ(5,101,END=315)CARD
CALL CRUNCH(ILNGTH)
IF(CARDDP(1).EQ.ALPHA(25)) GO TO 33
IF(CARDDP(1).EQ.ALPHA(14)) GO TO 313
315 WRITE(6,106)
GO TO 312
313 WRITE(6,105)
105 FORMAT('O',I0X,' IF YOU DESIRE TO HAVE A SMOOTH COPY OF YOUR',
* PROGRAM WITH ITS EXECUTION THEN REPLY :YES ; OTHERWISE
* REPLY : NO .')
20 READ(5,101,END=326)CARD
CALL CRUNCH(ILNGTH)
IF(CARDDP(1).NE.ALPHA(25))GO TO 25
PRINT SMOOTH COPY OF PROGRAM AND EXECUTE IT
C
C
C
314 WRITE(6,104)
104 FORMAT('O',/////////)
DO 13 I=1,NSTMT
103 WRITE(6,103)(SFIL(I,J),J=1,80)
135 FORMAT('O',I0X,80A1)
CALL INITIAL
136 DEBUG=0.0
INBIG=1
IARRAY=808
NFOR=0
INTERP=0
IEXERR=0
DO 15 K=1,NSTMT
DO 14 L=1,80
14 CARD(L)=SFIL(K,L)
15 CALL CRUNCH(ILNGTH)
CALL COMPLR(NFOR,IARRAY,ILNGTH,INBIG)
EXECUTION ERROR GO TO EDIT ROUTINE
IF(IEXERR.EQ.1)GO TO 31
WRITE(6,104)
GO TO 26
C
C
C

```





```

25 IF(CARDP(1).EQ.ALPHA(14))GO TO 26
326 WRITE(6,106)
106 FORMAT('O:'); *** YOUR REPLY IS INCORRECTLY TYPED REPLY AGAIN***//
      GO TO 20
26 WRITE(6,107)
107 FORMAT('I:'); 10X;' DO YOU WANT TO EXECUTE ANOTHER PROGRAM'/' REPLY
      * : YES OR NO.'//)
29 READ(5,101,END=320) CARD
      CALL CRUNCH(1,LENGTH)
      IF(CARDP(1).EQ.ALPHA(14))GO TO 30
      IF(CARDP(1).EQ.ALPHA(25))GO TO 3
330 WRITE(6,106)
      GO TO 29
30 RETURN

      EDIT ROUTINE.
31 WRITE(6,108)
108 FORMAT('O:'); 10X;' *** EXECUTION ERROR ***'
      * IF YOU WANT TO CORRECT YOUR PROGRAM NOW THEN REPLY : YES AND '//'
      * YOU WILL ENTER REPLY : NO.'//)
      * OTHERWISE REPLY : NO.'//)
208 READ(5,101,END=343)CARD
      CALL CRUNCH(1,LENGTH)
      IF(CARDP(1).NE.ALPHA(25)) GO TO 43
33 IF(OLD.EQ.1)GO TO 321
110 WRITE(6,110)
      * BY USING THE REFERENCE NUMBERS LISTED TO THE LEFT OF YOUR BASIC '//'
      * PROGRAM STATEMENT YOU MAY ADD, DELETE, OR CORRECT ONE LINE.'//
      * OF THE PROGRAM STATEMENT AT A TIME. IN ALL EDITING, THE FIRST STEP IS TO '//'
      * INPUT THE PROGRAM STATEMENT REFERENCE NUMBER AND HIT THE CARRIAGE '//'
      * / RETURN. THE SECOND STEP DEPENDS ON WHAT EDITING YOU DO : '//'
      * 1. DELETE STATEMENT REFERENCED IS DELETED BY TYPING THE '//'
      * THE BASIC CORRECT '//'
      * LETTERS CORRECT '//'
      * 2. CORRECT THE BASIC STATEMENT REFERENCED TYPE IN THE COMPLETE '//'
      * TO CORRECT BASIC STATEMENT. '//'
      * 3. ADD STATEMENT IS ADDED 'AFTER' THE BASIC STATEMENT '//'
      * REFERENCED BY TYPING THE LETTERS ADD1 FOLLOWED BY THE BASIC '//'
      * STATEMENT. ALL BLANKS FOLLOWING THE LETTERS ADD1 WILL BE '//'
      * INCLUDED IN THE BASIC STATEMENT. TO PLACE A STATEMENT 'BEFORE' '//'
      * / THE FIRST STATEMENT IN THE PROGRAM, USE THE REFERENCE NUMBER '//'
      * OLD=1 '//'

```

TES01640  
 TES01650  
 TES01660  
 TES01670  
 TES01680  
 TES01690  
 TES01710  
 TES01720  
 TES01730  
 TES01740  
 TES01750  
 TES01760  
 TES01770  
 TES01780  
 TES01790  
 TES01800  
 TES01810  
 TES01820  
 TES01830  
 TES01840  
 TES01850  
 TES01860  
 TES01870  
 TES01880  
 TES01890  
 TES01900  
 TES01910  
 TES01920  
 TES01930  
 TES01940  
 TES01950  
 TES01960  
 TES01970  
 TES01980  
 TES01990  
 TES02000  
 TES02010  
 TES02020  
 TES02030  
 TES02040  
 TES02050  
 TES02060  
 TES02070  
 TES02080  
 TES02090  
 TES02100  
 TES02110

C  
 C  
 C



```

C      WRITE PROGRAM WITH REFERENCE NUMBERS
C
321 DO 32 I=1,NSTMT
32 WRITE(6,109) I,(SFILE(I,J),J=1,80)
109 FORMAT(1,14,10X,80A1)
334 WRITE(6,111)
111 FORMAT(10,1) INPUT REFERENCE NUMBER NOW
335 KNT=0
    READ(5,101,END=331)CARD
    CALL CRUNCH(ILNGTH)
    IF(CARDP(1).EQ.ASTRSK) GO TO 35
    DO 237 I=1,2
    DO 235 J=1,10
C      CHECK FOR DIGIT
C
C      IF(CARDP(I).EQ.DIGIT(J)) GO TO 236
235 CCNTINUE
C      IF(J.EQ.10.AND.CARDP(I).EQ.BLANK) GO TO 237
331 WRITE(6,106)
    GO TO 34
C      CCNVERT ALPHA CHARACTER TO NUMBER
C
236 KNT=KNT+1
    IF(KNT.EQ.1) NREF=J-1
    IF(KNT.EQ.2) NREF=10*NREF+(J-1)
237 CONTINUE
C      CHECK FOR LEGAL REFERENCE NUMBER
C
C      IF(NREF.LE.NSTMT.AND.NREF.GE.0) GO TO 36
    WRITE(6,106)
    GO TO 35
36 WRITE(6,112)
112 FORMAT(10,1) INPUT EDITING NOW (DEL,BASIC STATEMENT,ADD1....)'/'
37 READ(5,101,END=36)CARD
    CALL CRUNCH(ILNGTH)
    IF(CARDP(1).EQ.ASTRSK) GO TO 36
C      DELETE CURRENT LINE
C
C      IF(CARDP(1).EQ.CMINUS) GO TO 50
C
C      ADD A STATEMENT AFTER CURRENT LINE
C
C      IF(CARDP(1).EQ.PLUS) GO TO 55

```

TES02120  
 TES02130  
 TES02140  
 TES02150  
 TES02160  
 TES02170  
 TES02180  
 TES02190  
 TES02200  
 TES02210  
 TES02220  
 TES02230  
 TES02240  
 TES02250  
 TES02260  
 TES02270  
 TES02280  
 TES02290  
 TES02300  
 TES02310  
 TES02320  
 TES02330  
 TES02340  
 TES02350  
 TES02360  
 TES02370  
 TES02380  
 TES02390  
 TES02400  
 TES02410  
 TES02420  
 TES02430  
 TES02440  
 TES02450  
 TES02460  
 TES02470  
 TES02480  
 TES02490  
 TES02500  
 TES02510  
 TES02520  
 TES02530  
 TES02540  
 TES02550  
 TES02560  
 TES02570  
 TES02580  
 TES02590



```

C      CORRECT CURRENT LINE
C
C      DEBUG=C.C
C      INTERP=1
C      IEXERR=0
C      DO 370 I=1,80
C      IF(CARDP(I).EQ.ALPHA(14).AND.CARDP(I+3).EQ.ALPHA(20)) GO TO 39
C      CCNTINUE
C
C      370
C      IF NO ERRORS , FILE INPUT STATEMENT
C
C      CALL CCPLR(NFOR,IARRAY,ILNGTH,INBIG)
C      IF(NERRS.EQ.0) GO TO 39
C      NERRS=0
C      GO TO 34
C      39 DO 40 J=1,80
C      40 SFILE(NREF,J)=CARD(J)
C      GO TO 47
C      43 IF(CARDP(1).EQ.ALPHA(14)) GO TO 44
C      343 WRITE(6,106)
C      GO TO 208
C      44 WRITE(6,113)
C      113 FORMAT('0',10X,' IF YOU DESIRE TO HAVE THE PROGRAM RUN WITH ',//
C      *THE DEBUG FEATURE REPLY: YES ; OTHERWISE REPLY: NO AND THIS ',//
C      *PROGRAM WILL BE LOST .')
C      CALL CRUNCH(ILNGTH)
C      45 READ(5,101,END=346)CARD
C      IF(CARDP(1).NE.ALPHA(25)) GO TO 46
C      EXECUTE WITH DEBUG FEATURE ADDED
C
C      CALL INITAL
C      DEBUG=1.0
C      GO TO 136
C      46 IF(CARDP(1).EQ.ALPHA(14)) GO TO 26
C      346 WRITE(6,106)
C      GO TO 45
C      47 WRITE(6,114)
C      114 FORMAT('0',10X,' MORE CORRECTIONS TO BE MADE ?? REPLY: YES ',//
C      *OR REPLY: NO AND THE EDITED PROGRAM WILL BE EXECUTED .')
C      48 READ(5,101,END=348)CARD
C      CALL CRUNCH(ILNGTH)
C      IF(CARDP(1).EQ.ALPHA(25)) GO TO 321
C      IF(CARDP(1).EQ.ALPHA(14)) GO TO 314
C      348 WRITE(6,106)
C      GO TO 48
C
C
C

```



|     |                                                                 |          |
|-----|-----------------------------------------------------------------|----------|
| C   | DELETE STATEMENT                                                | TES03080 |
| C   |                                                                 | TES03090 |
| 50  | NSTMT=NSTMT-1                                                   | TES03100 |
|     | DC 52 I=NREF,NSTMT                                              | TES03110 |
|     | DO 52 J=1,80                                                    | TES03120 |
| 52  | SFILE(I,J)=SFILE(I+1,J)                                         | TES03130 |
|     | GO TO 47                                                        | TES03140 |
| C   |                                                                 | TES03150 |
| C   | ADD A BASIC STATEMENT AFTER CURRENT LINE                        | TES03160 |
| C   |                                                                 | TES03170 |
| 55  | IF(NSTMT.GT.99) GO TO 65                                        | TES03180 |
|     | NSTMT=NSTMT +1                                                  | TES03190 |
|     | ITEMP=NSTMT                                                     | TES03200 |
|     | DEBUG=0.0                                                       | TES03210 |
|     | INTERP=1                                                        | TES03220 |
|     | IEXERR=0                                                        | TES03230 |
| C   |                                                                 | TES03240 |
| C   | FIND BEGINNING OF STATEMENT                                     | TES03250 |
| C   |                                                                 | TES03260 |
|     | DO 56 K=1,80                                                    | TES03270 |
|     | IF(CARD(K).EQ.BLANK) GO TO 56                                   | TES03280 |
|     | IF(CARD(K).NE.ALPHA(1)).OR.CARD(K+2).NE.ALPHA(4)) GO TO 56      | TES03290 |
|     | IF(CARD(K+3).EQ.DIGIT(2))GO TO 60                               | TES03300 |
| 56  | CONTINUE                                                        | TES03310 |
|     | WRITE(6,106)                                                    | TES03320 |
|     | GO TO 37                                                        | TES03330 |
| 60  | IBEG=K+3                                                        | TES03340 |
|     | DO 600 L=1,80                                                   | TES03350 |
|     | IBEG=IBEG+1                                                     | TES03360 |
|     | IF(IBEG.GT.80) GO TO 601                                        | TES03370 |
|     | CARD(L)=CARD(IBEG)                                              | TES03380 |
| 600 | CONTINUE                                                        | TES03390 |
| 601 | CALL CRUNCH(ILNGTH)                                             | TES03400 |
|     | IF(CARDP(1).EQ.ASTRSK) GO TO 36                                 | TES03410 |
|     | DO 603 I=1,80                                                   | TES03420 |
|     | IF(CARDP(I).EQ.ALPHA(14).AND.CARDP(I+3).EQ.ALPHA(20)) GO TO 602 | TES03430 |
| 603 | CONTINUE                                                        | TES03440 |
|     | CALL COMPLR(NFOR,IARRAY,ILNGTH,INBIG)                           | TES03450 |
| C   |                                                                 | TES03460 |
| C   | IF NO ERRORS THEN FILE STATEMENT                                | TES03470 |
| C   |                                                                 | TES03480 |
|     | IF(NERRS.EQ.0) GO TO 602                                        | TES03490 |
|     | NERRS=0                                                         | TES03500 |
|     | GO TO 36                                                        | TES03510 |
| C   |                                                                 | TES03520 |
| C   | ADD STATEMENT TO PROGRAM FILE. MOVE STATEMENTS UP AND DOWN      | TES03530 |
| C   |                                                                 | TES03540 |
| 602 | IADD=NREF+1                                                     | TES03550 |









```

C      CHECK FOR END OF SESSION (QUIT)
C      CRU00370
C      CRU00380
C      CRU00390
C      CRU00400
C      CRU00410
C      CRU00420
C      CRU00430
C      CRU00440
C      CRU00450
C      CRU00460
C      CRU00470
C      CRU00480
C      CRU00490
C      CRU00500
C      CRU00510
C      CRU00520
C      CRU00530
C      CRU00540

26  IF(CARDP(1).NE.ALPHA(17).OR.CARDP(I+1).NE.ALPHA(21)) GO TO 27
    IF(CARDP(I+2).EQ.ALPHA( 9).AND.CARDP(I+3).EQ.ALPHA(20)) STOP

C      CHECK FOR EDIT COMMAND (ADD1)
C      CRU00430
C      CRU00440
C      CRU00450
C      CRU00460
C      CRU00470
C      CRU00480
C      CRU00490
C      CRU00500
C      CRU00510
C      CRU00520
C      CRU00530
C      CRU00540

27  IF(CARDP(1).NE.ALPHA(1).OR.CARDP(I+1).NE.ALPHA(4))GO TO 28
    IF(CARDP(I+2).EQ.ALPHA(4).AND.CARDP(I+3).EQ.DIGIT(2))CARDP(1)=PLUS

C      CHECK FOR EDIT COMMAND (DEL)
C      CRU00430
C      CRU00440
C      CRU00450
C      CRU00460
C      CRU00470
C      CRU00480
C      CRU00490
C      CRU00500
C      CRU00510
C      CRU00520
C      CRU00530
C      CRU00540

28  IF(CARDP(1).NE.ALPHA(4).OR.CARDP(I+1).NE.ALPHA(5)) GO TO 25
    IF(CARDP(I+2).EQ.ALPHA(12))CARDP(1)=CMINUS
CONTINUE
RETURN
END

```

#### SUBROUTINE INITIAL

```

C      THIS ROUTINE INITIALIZES THE SYSTEM. IT LOADS ALL VECTORS
C      WITH APPROPRIATE INFORMATION TO BE USED BY THE SYSTEM. IT ALSO
C      SETS CERTAIN POINTERS TO THEIR INITIAL VALUES.
C      CRU00110
C      CRU00180
C      CRU00190
C      CRU00200
C      CRU00210
C      CRU00220
C      CRU00230
C      CRU00240
C      CRU00250
C      CRU00260
C      CRU00270
C      CRU00280
C      CRU00290

```

```

COMMON
-  STACK(100), PROG(2000), CARD(80), CARDP(80), ALPHA(48),
-  IAPTR, INPTR, IADATA(500), XNDATA(500), STRING(5),
-  DIGIT(10), IPRIB(10), LISTST(100), ISTLST(100),
-  PRT(2500), NERRS, INST, NSTLST, DEBUG, DOLSGN, QUOTE,
-  EQUALS, PARRT, DECIMAL, PLUS, CMINUS, SLASH, COMMA,
-  PARLET, ASTRSK, BLANK
COMMON /RANDOMM/IRAND
DIMENSION ATEMP(26), DIGTMP(10), CHARTM(12)
DATA ATEMP/1HA,1HB,1HC,1HD,1HE,1HF,1HG,1HH,1HI,1HJ,
1HK,1HL,1HM,1HN,1HO,1HP,1HQ,1HR,1HS,1HT,1HU,1HV,1HW,1HX,1HY,1HZ/
DATA DIGTMP/1H0,1H1,1H2,1H3,1H4,1H5,1H6,1H7,1H8,1H9/
DATA CHARTM/1H+,1H-,1H.,1H=,1H!,1H',1H$,
1H~,1H//
INITIALIZE
IRAND=5555
NERRS=0
INST=0
NSTLST=0

```



|   |                                   |          |
|---|-----------------------------------|----------|
| C | IAPTR=C                           | INI00290 |
| C | INPTR=0                           | INI00300 |
|   | INITIALIZE PRIORITY TABLE         | INI00310 |
|   | IPRITB(1)=2                       | INI00320 |
|   | IPRITB(2)=2                       | INI00330 |
|   | IPRITB(3)=3                       | INI00340 |
|   | IPRITB(4)=3                       | INI00350 |
|   | IPRITB(5)=4                       | INI00360 |
|   | IPRITB(6)=5                       | INI00370 |
| C | SET UP VOCABULARY                 | INI00380 |
| C | LOAD ALPHABET                     | INI00390 |
| C |                                   | INI00400 |
|   | DO 10 I=1,26                      | INI00410 |
|   | 10 ALPHA(I)=ATEMP(I)              | INI00420 |
| C | LOAD DIGITS                       | INI00430 |
| C |                                   | INI00440 |
| C |                                   | INI00450 |
|   | DO 20 I=1,10                      | INI00460 |
|   | 20 DIGIT(I)=DIGTMP(I)             | INI00470 |
| C | LOAD DIGITS INTO ALPHA            | INI00480 |
| C |                                   | INI00490 |
| C |                                   | INI00500 |
|   | LOC=26                            | INI00510 |
|   | DC 12 I=1,10                      | INI00520 |
|   | LCC=LOC+1                         | INI00530 |
|   | 12 ALPHA(LOC)=DIGIT(I)            | INI00540 |
| C | LOAD SPECIAL CHAR INTO ALPHA      | INI00550 |
| C |                                   | INI00560 |
| C |                                   | INI00570 |
|   | DO 16 I=1,12                      | INI00580 |
|   | LCC=LOC+1                         | INI00590 |
|   | 16 ALPHA(LOC)=CHARTM(I)           | INI00600 |
| C | INITIALIZE ARRAY STORAGE POINTERS | INI00610 |
| C |                                   | INI00620 |
| C |                                   | INI00630 |
|   | DO 50 I=287,384,4                 | INI00640 |
|   | 50 PRT(I)=0.0                     | INI00650 |
| C | LOAD SPECIAL CHARACTERS           | INI00680 |
| C |                                   | INI00690 |
| C |                                   | INI00700 |
|   | ASTRSK=CHARTM(1)                  | INI00710 |
|   | BLANK =CHARTM(2)                  | INI00720 |
|   | COMMA =CHARTM(3)                  | INI00730 |
|   | DECIMAL=CHARTM(4)                 | INI00740 |
|   | EQUALS=CHARTM(5)                  | INI00750 |
|   | PARRT =CHARTM(6)                  | INI00760 |









```

*BY AN END STATEMENT. THE WORDS : REM , READ, LET , PRINT , DATA,
*//, AND END ARE KEY WORDS THAT MAKE UP A BASIC STATEMENT. //)
  READ(5,101,END=304)CARD
304 WRITE(6,104)
104 FORMAT(0,10X, ' MOST OF THE KEY WORDS USED IN THE BASIC
    STATEMENTS ARE SELF-EXPLANATORY:
    REM - REMARKS/COMMENTS
    READ M,G - ASSIGNS NUMBERS IN THE DATA STATEMENT TO THE
    LET - ASSIGNS THE RESULT OF M DIVIDED BY G INTO VARIABLE T
    PRINT - PRINTS THE STRING IN SINGLE QUOTES TO BE
    PRINTED LITERALLY
    PRINT M,G,T - PRINTS THE CURRENT VALUE OF THE VARIABLES M,G,T, //5X,
    END TELLS THE COMPUTER THAT THE INPUT PROGRAM IS TO BE
    EXECUTED.//)
  READ(5,101,END=305)CARD
305 WRITE(6,105)
105 FORMAT(0,10X, ' IF AT THIS POINT YOU WOULD LIKE TO RUN THE
    SAMPLE PROGRAM, TO GAIN SOME CONFIDENCE IN THE COMPUTER AND ITS
    ABILITY TO PROVIDE SPEEDY RESULTS, THEN REPLY : YES ; OTHER-
    WISE REPLY : NO AND THE INSTRUCTION WILL CONTINUE. //)
10 READ(5,101,END=306)CARD
    CALL CRUNCH(LENGTH)
    IF(CARDP(1).EQ.ASTRSK) GO TO 10
    IF(CARDP(1).EQ.ALPHA(14)) GO TO 25
    IF(CARDP(1).EQ.ALPHA(25)) GO TO 15
306 WRITE(6,106)
106 FORMAT(0,10X, ' *** YOUR REPLY WAS TYPED INCORRECTLY ; CHECK THE
    QUESTION AND REPLY AGAIN *** //)
    GO TO 10

EXECUTE SAMPLE PROGRAM

15 WRITE(6,107)
107 FORMAT(0,10X, ' TO EXECUTE THE SAMPLE PROGRAM TYPE IN THE BASIC
    STATEMENTS AS THEY APPEAR, AND DONT WORRY ABOUT SPACING. THE
    COMPUTER ANALIZES EACH BASIC STATEMENT AS IT IS INPUT ; AND
    IF THERE ARE NO ERRORS, IT UNLOCKS THE KEYBOARD AND WAITS FOR
    YOUR NEXT INPUT. WHEN THE END STATEMENT IS INPUT THE PROGRAM
    IS EXECUTED. //)
  READ(5,101,END=308)CARD
308 WRITE(6,108)
108 FORMAT(0,10X, ' HOWEVER, IF YOU MAKE AN ERROR THE COMPUTER WILL
    TELL YOU THE ERROR, AND EXPECT A CORRECTION. DONT WORRY ABOUT
    THE ERROR CHECK YOUR INPUT AGAINST THE SAMPLE PROGRAM, AND
    INPUT FOUR ASTERIXES(****), HIT RETURN, ANY PAIR OF INTEGER OR DECIMAL
    CORRECT STATEMENT. //10X, ' YOU MAY USE ANY PAIR OF INTEGER OR DECIMAL

```

LES00380  
 LES00390  
 LES00400  
 LES00410  
 LES00420

LES00500  
 LES00510  
 LES00520  
 LES00530  
 LES00540  
 LES00550  
 LES00560  
 LES00570  
 LES00580  
 LES00590  
 LES00600  
 LES00610  
 LES00620  
 LES00630  
 LES00640  
 LES00650  
 LES00660  
 LES00670  
 LES00680  
 LES00690  
 LES00700  
 LES00710  
 LES00720  
 LES00730  
 LES00740  
 LES00750  
 LES00760  
 LES00770  
 LES00780  
 LES00790  
 LES00800  
 LES00810  
 LES00820

C  
 C  
 C



```

* MAL NUMBERS: // FOR INPUT DATA YOU MAY OMIT THE REM STATEMENTS AND
* THE PRINT // MILES TRAVELED: // // // ETC. // STATEMENT IF YOU DONT WANT
* T TO TYPE A LOT OF STATEMENTS: // //
CALL TEST1
25 WRITE(6,111)
111 FORMAT(1X,111)
* NE, FOLLOWED BY AN END STATEMENT. // BECAUSE NO BASIC STATEMENT
* MAY BE LONGER THAN ONE INPUT LINE (80 SPACES), // THERE IS NO PR
* DIVISION FOR CONTINUING STATEMENTS FROM ONE LINE // TO THE NEXT.
* HOWEVER: YOU MAY SPACE THE INPUT LINE // AS DESIRED FOR READABILITY
* TY SINCE THE COMPUTER IGNORES BLANKS IN BASIC. // //
READ(5,101,END=309) CARD
309 WRITE(6,112)
112 FORMAT(1X,10X,1)
* STATEMENT NUMBERS 1. EACH BASIC STATEMENT MAY HAVE AN OPTIONAL
* STATEMENT NUMBER PRECEDING IT FOR IDENTIFICATION PURPOSES.
* THIS STATEMENT NUMBER MUST BE AN INTEGER BETWEEN 1 -> 9999
* 5X, // FOR EXAMPLE: 12 READ M,G
* 10X, // 2. KEY WORDS // THE KEY WORDS THAT MAKE UP A
* BASIC STATEMENT (REM, READ, LET, ETC.) // ARE SPECIAL TERMINAL SYM
* BOLS THAT ARE RECOGNIZED BY THE COMPUTER. // AND FOR THIS REASON TH
* KEY MUST BE SHELLED CORRECTLY AND ONLY // USED IN BASIC STATEMENTS
* // 10X, // THE END STATEMENT INDICATES THAT THE INPUT PROGRAM IS //
* COMPLETED AND THAT PROGRAM EXECUTION IS TO BEGIN. THE //
* STATEMENT IS ALWAYS THE LAST STATEMENT IN A PROGRAM. // //
READ(5,101,END=313) CARD
313 WRITE(6,113)
113 FORMAT(1X,113)
* WHAT YOU HAVE JUST LEARNED. // //
WRITE(6,114)
114 FORMAT(1X,10X,1)
* NO // 5X, // REM ONE LINE DO NOTHING PROGRAM // 5X, // END // //
30 READ(5,101,END=315) CARD
CALL CRUNCH(1,LENGTH)
IF(CARDP(1).EQ.ASTRSK) GO TO 30
IF(CARDP(1).EQ.ALPHA(14).OR.CARDP(1).EQ.ALPHA(25)) GO TO 31
315 WRITE(6,106)
GO TO 30
31 WRITE(6,115)
115 FORMAT(1X,115)
* YES, THE SIMPLEST BASIC PROGRAM CONSISTS OF JUST AN
* STATEMENT. //
WRITE(6,116)
116 FORMAT(1X,10X,1)
* WHICH OF THE FOLLOWING BASIC STATEMENTS IS //
* IN THE PROPER FORMAT: // 15X, // A. // 15 LET T=M/G // 15X, // B. // 15 LET
* M/G // 15X, // C. // 15 LET T = M / G // 10X, // REPLY A, B, C OR
* ALL. // //
32 READ(5,101,END=317) CARD

```



```

CALL CRUNCH(ILNGTH)
IF(CARDP(1).EQ.ASTRSK) GO TO 32
IF(CARDP(1).EQ.ALPHA(1).OR.CARDP(1).EQ.ALPHA(2)) GO TO 33
IF(CARDP(1).EQ.ALPHA(3)) GO TO 33
317 WRITE(6,106)
GO TO 32
33 WRITE(6,117)
117 FORMAT(0), ALL OF THE ABOVE BASIC STATEMENTS ARE CORRECT BECAUSE
*// SPACES ARE DISREGARDED. NOTE THAT STATEMENT B.), ALTHOUGH//
*// CORRECT IS CONFUSING TO READ. BLANKS AND INDENTATIONS MAKE A//
*// PROGRAM EASY TO READ.//)
218 WRITE(6,218)
FORMAT(0), 5X, C. ALPHA-NUMERIC CHARACTERS // 5X,
*// ALPHA-NUMERIC CHARACTERS ARE THE LEGAL CHARACTERS, DIGITS, AND//
*// SPECIAL CHARACTERS THAT CAN BE USED IN BASIC. //10X, //
*// 1. CHARACTERS CONSIST OF THE LETTERS IN THE ALPHABET A-->Z //10X,
*// 2. DIGITS ARE THE SINGLE NUMBERS 0-->9 //10X,
*// 3. SPECIAL CHARACTERS ARE ** / + - ( ) = ' " $ % //10
*// 4. A STRING IS ANY LIST OF ALPHA-NUMERIC CHARACTERS.
*// ENCLOSED IN SINGLE QUOTES. FOR EXAMPLE: //5X, "THIS IS A
*// STRING." //)
READ(5,101,END=318)CARD
318 WRITE(6,118)
118 FORMAT(0), 5X, D. VARIABLES
*// 10X, IN BASIC, THERE ARE THREE TYPES OF VARIABLES: SIMPLE, ALPHA
*// AND SUBSCRIPTED. SIMPLE VARIABLES ARE IDENTIFIED BY A SINGLE LETTER
*// LESSON 5. //10X, 1. SIMPLE VARIABLES FOLLOWED BY A DIGIT BETWEEN 0-->9
*// OR A SINGLE LETTER FOLLOWED BY A DIGIT BETWEEN 0-->9. BUT,
*// FOR EXAMPLE: A, A3, Z0, AND X ARE LEGAL VARIABLES; BUT,
*// 20X, A26, 9Z, ABC, AND X1Y ARE ILLEGAL VARIABLES. // THEREF
*// YOU HAVE 286 SIMPLE VARIABLES FOR USE IN YOUR PROGRAMS. //)
READ(5,101,END=319)CARD
319 WRITE(6,119)
119 FORMAT(0), 10X, 2. ALPHA VARIABLES
*// ALPHA VARIABLES ARE USED FOR ALPHA-NUMERIC MANIPU
*// IN WHICH A GROUP OF ALPHA-NUMERIC CHARACTERS, CALLED//
*// A STRING, ARE REPRESENTED BY AN ALPHA VARIABLE. THE ALPHA//
*// VARIABLE CONSISTS OF A SINGLE LETTER FOLLOWED BY A DOLLAR SIGN,
*// THE MAXIMUM LENGTH IS 16 CHARACTERS.//
*// TO THE ALPHA VARIABLE IS A$, X$, Z$. ARE LEGAL ALPHA VARIABLES.//
*// FOR EXAMPLE: A$, X$, Z$. ARE LEGAL ALPHA VARIABLES.//
*// A SIMPLE USE OF ALPHA VARIABLES: READ A$,B$,// 5X//
*// PRINT A$,B$,// 5X, "MONDAY",// 21 JUNE,// 5X, "END",//, TH
*// PROGRAM WILL PRODUCE THE OUTPUT: //10X, "MONDAY 21 JUNE"//)

```





```

320 READ(5,101,END=320)CARD
120 WRITE(6,120) 5X,'E. NUMBERS
    *10X,' NUMBERS MAY BE EXPRESSED AS INTEGERS OR AS REALS ,AND A
    **NUMBER WHETHER INTEGER OR REAL IS LIMITED TO 9 DIGITS ,NOT INCLUD
    ***ING./, DECIMAL POINT. A NUMBER IS ASSUMED POSITIVE UNLESS IT
    **IS PRECEDED BY A - SIGN .
    *10X,' 1. INTEGERS
    **RS ARE NUMBERS WITH NO FRACTIONAL PART ,IE. 3 , 15
    *10X,' 2. REALS
    **REAL (FIXED-POINT) NUMBERS HAVE A DECIMAL POINT AND A FRACTIONAL
    **PART ,IE. 3.0 , 15.31, 729.1 , 0.0 (//)
    READ(5,101,END=321)CARD
321 WRITE(6,121) 5X,' F. EXPRESSIONS
121 FORMAT(0),CN MAY BE A SINGLE CONSTANT OR VARIABLE , OR AN
    *AN EXPRESSION ,CN MAY BE A SINGLE CONSTANT OR VARIABLE , OR AN
    **ARITHMETIC EXPRESSION ,CN MAY BE A SINGLE CONSTANT OR VARIABLE , OR AN
    ***USING OPERATORS AND PARENTHESIS ,//10X,' 1. OPERATORS
    *10X,' NUMBERS AND SIMPLE VARIABLES MAY BE COMBINED INTO ARITHMETIC
    **EXPRESSIONS BY USING ONE OR MORE OF THE ARITHMETIC OPERATORS:
    ***//4X,' ** EXPONENTIATION ,/5X,' * MULTIPLICATION ,/5X,' / DIVISION
    *ON ,/5X,' + ADDITION ,/5X,' - SUBTRACTION
    **IN WRITING AN EXPRESSION , EACH ARITHMETIC OPERATION MUST BE SHOWN
    ***. NO TWO OPERATIONS MAY BE ADJACENT ,//10X,' 2. PARENTHESES
    *VARIABLES BE ADJACENT IN AN EXPRESSION ,//10X,' 3. PARENTHESES
    **B+C ,X*Z , AND Y-3.0 ARE LEGAL EXPRESSIONS. (//)
    READ(5,101,END=322)CARD
322 WRITE(6,122)
122 FORMAT(0),10X,' 2. PARENTHESES
    *// TO CONTROL THE ORDER IN WHICH AN ARITHMETIC EXPRESSION IS EVALU
    **ATED .// THE NORMAL HIERARCHY OF OPERATORS IS : /5X,' 1. *//5X,
    ***2. PARENTHESIS ARE USED , AND EXPRESSION WITHIN A PARENTHESIS PAIR
    **// IS EVALUATED FIRST .// PARENTHESIS IN EXPRESSIONS MUST ALWAYS
    ***CCUR IN PAIRS ,IE. THE VALUE 5*(4+3) , (5*(4+3)) ARE EQUIVALENT
    **OPERATIONS ARE EVALUATED ACCORDING TO THE OPERATORS HIERARCHY.//
    ***IF TWO OPERATORS OF THE SAME HIERARCHY OCCUR IN AN EXPRESSION ,//
    **EVALUATION IS FROM LEFT TO RIGHT .// FOR EXAMPLE: 2*4 + 6/3 ,//
    ***IN THIS EXPRESSION 2*4 WILL BE EVALUATED FIRST , THEN 6/3 ,//
    **AND THEN THE TWO RESULTS WILL BE ADDED .(//)
    READ(5,101,END=323)CARD
323 WRITE(6,123)
123 FORMAT(0),10X,' 1.) 4 + 6 / 2 HAS THE VALUE (REPLY WITH VALUE) (//
    *//)
34 READ(5,101,END=324)CARD

```





```

35 CALL CRUNCH(ILNGTH)
36 IF(CARDP(1).EQ.ASTRSK) GO TO 34
37 DO 35 I=1,10
38 IF(CARDP(1).EQ.DIGIT(I)) GO TO 36
39 CCNTINUE
324 WRITE(6,106)
124 GO TO 34
124*D AS FOLLOWS :
36 WRITE(6,124)
125 WRITE(6,125)
126 WRITE(6,126)
127 WRITE(6,127)
128 READ(5,101,END=327)CARD
39 CALL CRUNCH(ILNGTH)
40 IF(CARDP(1).EQ.ASTRSK) GO TO 38
41 IF(CARDP(1).EQ.DIGIT(6)) GO TO 41
42 DO 39 I=1,10
43 IF(CARDP(1).EQ.DIGIT(I)) GO TO 40
44 CCNTINUE
327 WRITE(6,106)
40 WRITE(6,124)
127 WRITE(6,127)
41 WRITE(6,128)
128* /
42 READ(5,101,END=46)CARD
43 CALL CRUNCH(ILNGTH)
44 IF(CARDP(1).EQ.ASTRSK) GO TO 42
45 IF(CARDP(1).EQ.DIGIT(3).AND.CARDP(2).EQ.DIGIT(6)) GO TO 48
46 DO 43 I=1,10
47 IF(CARDP(1).EQ.DIGIT(I)) GO TO 44
48 CCNTINUE
43 GO TO 46
44 CCNTINUE
45 DO 45 J=1,10
46 IF(CARDP(2).EQ.DIGIT(J)) GO TO 47
47 CCNTINUE
48 WRITE(6,106)
49 GO TO 42
50 WRITE(6,124)
51 WRITE(6,129)
52 FORMAT(0,10X, ' ((4+6)/2)**2 -> (10/2)**2 -> (5)**2 -> 25' /)
53 WRITE(6,130)

```

```

LES02270
LES02280
LES02290
LES02300
LES02310
LES02320
LES02330
LES02340
LES02350
LES02360
LES02370
LES02380
LES02390
LES02400
LES02410
LES02420
LES02430
LES02440
LES02450
LES02460
LES02470
LES02480
LES02490
LES02500
LES02510
LES02520
LES02530
LES02540
LES02550
LES02560
LES02570
LES02580
LES02590
LES02600
LES02610
LES02620
LES02630
LES02640
LES02650
LES02660
LES02670
LES02680
LES02690
LES02700
LES02710
LES02720
LES02730
LES02740

```



```

130 FORMAT('O', 5X, ' F. SUMMARY
*// YOU WILL NOW BE GIVEN A SAMPLE THE INSTRUCTION SET FOR LESSON 1.//
**// THE MISTAKES IN IT, CONCERNING WHAT YOU HAVE LEARNED IN THIS.//
**// LESSON : //5X, 1, REM REVIEW PROGRAM, //5X, 2, READ A1,B2,Z1,$
**Y//5X, 3, LET A1=B2(A1+Z1)//6X, 4, LET B2=((A1*3)+Z1**2)//5X, 5
** LET Z1=3.141592763//5X, 6, PRINT 1A,B2,Z1//5X, 7, DATA 5.0,3.3.
**1416, //FINI.//. AFTER LOOKING AT THE SAMPLE PROBLEM, YOU WILL B
**E ASKED QUESTIONS.//. ABOUT EACH STATEMENT, //)
  READ(5,101,END=331)CARD
331 WRITE(6,131)
131 FORMAT('O', 10X, ' 1.) LINE 2 CONTAINS A READ STATEMENT FOLLOWED'//
* BY A LIST OF VARIABLES. ANY ERRORS (REPLY: YES OR NO)?'//)
50 READ(5,101,END=332)CARD
  CALL CRUNCH(ILNGTH)
  IF(CARDP(1)).EQ.ASTRSK) GO TO 50
  IF(CARDP(1)).EQ.ALPHA(14)) GO TO 52
  IF(CARDP(1)).EQ.ALPHA(25)) GO TO 51
332 WRITE(6,106)
  GO TO 50
51 WRITE(6,132)
132 FORMAT('O', 10X, ' INPUT CORRECTION TO THE INCORRECT VARIABLE ONLY .'//)
510 READ(5,101,END=52)CARD
  CALL CRUNCH(ILNGTH)
  IF(CARDP(1)).EQ.ASTRSK) GO TO 51C
  IF(CARDP(1)).EQ.ALPHA(25).AND.CARDP(2).EQ.ALPHA(46)) GO TO 53
52 WRITE(6,133)
133 FORMAT('O', 10X, ' THE ILLEGAL VARIABLE IS $Y, SIMPLE VARIABLES ARE'//
* $, // ARE A LETTER OR A LETTER FOLLOWED BY A SINGLE DIGIT; ALPHA VARIABLE
* $, // ARE A LETTER FOLLOWED BY A DOLLAR SIGN; $, IE. Y$.')
53 WRITE(6,144)
144 FORMAT('O', 10X, ' 2.) LINE 3 CONTAINS A LET STATEMENT FOLLOWED'//
* BY A1 = EXPRESSION. ANY ERRORS (REPLY: YES OR NO)?'//)
54 READ(5,101,END=345)CARD
  CALL CRUNCH(ILNGTH)
  IF(CARDP(1)).EQ.ASTRSK) GO TO 54
  IF(CARDP(1)).EQ.ALPHA(14))GO TO 58
  IF(CARDP(1)).EQ.ALPHA(25))GO TO 55
345 WRITE(6,106)
  GO TO 54
55 WRITE(6,134)
134 FORMAT('O', 10X, ' INPUT CORRECTION TO ILLEGAL EXPRESSION; EVERYTHING'//
* AFTER = SIGN .'//)
56 READ(5,101,END=58)CARD
  CALL CRUNCH(ILNGTH)
  IF(CARDP(1)).EQ.ASTRSK) GO TO 56
  IF(CARDP(1)).EQ.ALPHA(2).AND.CARDP(2).EQ.DIGIT(3)) GO TO 57
  IF(CARDP(3)).NE.ASTRSK.OR.CARDP(4).NE.PARLFT)GO TO 58
57

```



```

IF(CARDP(5).NE.ALPHA(1).OR.CARDP(6).NE.DIGIT(2))GO TO 58
IF(CARDP(7).NE.PLUS.OR.CARDP(8).NE.ALPHA(26))GO TO 58
IF(CARDP(9).EQ.DIGIT(2).AND.CARDP(10).EQ.PARKT)GO TO 59
58 WRITE(6,135)
135 *EACH ARITHMETIC OPERATION MUST BE WRITTEN OUT ; A(B) IS NOT ASSUMED, //, TO BE A*(B) .,
59 WRITE(6,136)
136 *REPLY YES OR NO) ?(//)
60 READ(5,101,END=361)CARD
CALL CRUNCH(1)EQ.ASTRSK) GO TO 60
IF(CARDP(1).EQ.ALPHA(25))GO TO 64
IF(CARDP(1).EQ.ALPHA(14)) GO TO 55
361 GO TO 60
64 WRITE(6,137)
137 *THE EXPRESSION IS CORRECT .,
65 WRITE(6,138)
138 *BY A NUMBER . ANY ERRORS (REPLY : YES OR NO) ?(//)
70 READ(5,101,END=371)CARD
CALL CRUNCH(1)EQ.ASTRSK) GO TO 70
IF(CARDP(1).EQ.ALPHA(14).OR.CARDP(1).EQ.ALPHA(25)) GO TO 71
371 GO TO 70
71 WRITE(6,139)
139 *THE NUMBER CONTAINS 10 DIGITS AND THE MAXIMUM ALLOWED //, IS 9 DIGITS.,
140 *WRITE(6,140)
72 *BY A LIST OF VARIABLES . ANY ERRORS (REPLY : YES OR NO) ?(//)
72 READ(5,101,END=373)CARD
CALL CRUNCH(1)EQ.ASTRSK) GO TO 72
IF(CARDP(1).EQ.ALPHA(14)) GO TO 74
IF(CARDP(1).EQ.ALPHA(25)) GO TO 73
373 GO TO 72
73 WRITE(6,132)
730 *READ(5,101,END=74)CARD
CALL CRUNCH(1)EQ.ASTRSK) GO TO 730
IF(CARDP(1).EQ.ALPHA(1).AND.CARDP(2).EQ.DIGIT(2)) GO TO 75
74 WRITE(6,141)
141 *THE ILLEGAL VARIABLE IS 1A . SIMPLE VARIABLES ARE A'//

```

```

LES03230
LES03240
LES03250
LES03260
LES03270
LES03280
LES03290
LES03300
LES03310
LES03320
LES03330
LES03340
LES03350
LES03380
LES03390
LES03510
LES03540
LES03550
LES03560
LES03570
LES03580
LES03590
LES03600
LES03610
LES03620
LES03630
LES03640
LES03650
LES03660
LES03670
LES03680
LES03690
LES03700
LES03710
LES03720
LES03730
LES03740
LES03750
LES03760
LES03770
LES03780
LES03790
LES03800
LES03810
LES03820

```





```

* , LETTER OR A LETTER FOLLOWED BY A SINGLE DIGIT , IE. A1 .')
75 WRITE(6,142)
142 FORMAT(20,10X,' THERE ARE NO ERRORS IN LINE 7 . IS THE PROGRAM'//,
* READY TO EXECUTE (ASSUMING ABOVE ERRORS CORRECTED)'//, (REPLY : YE
* S OR NO)'//)
750 READ(5,101,END=80)CARD
CALL CRUNCH(1,LENGTH)
IF(CARDP(1).EQ.ASTRSK) GO TO 750
IF(CARDP(1).EQ.ALPHA(14).OR.CARDP(1).EQ.ALPHA(25)) GO TO 80
WRITE(6,106)
80 WRITE(6,143)
143 FORMAT(20,10X,' THE PROGRAM FORMAT REQUIRES THAT AN END STATEMENT
* , BE THE LAST STATEMENT OF THE PROGRAM . THUS THIS SAMPLE PROGRAM
* , WOULD NOT EXECUTE .'//, THIS CONCLUDES THE REVIEW QUESTIONS'//
* , FROM LESSON 1 .'//)
CALL EXIT
END

```

```

LESSON 2 INTRODUCES THE REM , PRINT , READ , DATA , RESTORE ,
AND RESTORE$ STATEMENTS
COMMON
STACK(100), PROG(2000), CARD(80), CARDP(80), ALPHA(48),
IAPTR, INPTR, IADATA(500), XNDATA(500), STRING(5),
DIGIT(10), IPRTIB(10), LISIST(100), ISTLST(100),
PRT(2500), NERRS, INST, NSIIST, DEBUG, DOLSGN, QUOTE,
EQUALS, PARRT, DECIMAL, PLUS, CMINUS, SLASH, COMMA,
PARLFT, ASTRSK, BLANK
CCMMCN INTERP, IEXERR, /
REAL*8 LES2, LESCN2
WRITE(6,100)
100 FORMAT(6,10X,' ** LESSON 2. **
* THIS INSTRUCTION SEQUENCE WILL INTRODUCE YOU TO THE BASIC LANGUAGE
* , STATEMENTS REM , PRINT , READ , AND DATA , USING THESE STATEMENTS
* , YOU WILL BE ABLE TO CONSTRUCT AND EXECUTE ELEMENTARY PROGRAMS
* , THE FORM FOR EACH BASIC STATEMENT WILL INCLUDE :
* , KEY WORD , << ELEMENTS OR LIST OF ELEMENTS SEPARATED BY COMMA
* , >>
* THE CARET SYMBOLS '<<<>>' DELINIATE THE LEGAL ITEMS THAT MAY
* FOLLOW THE KEYWORD AND MAKE UP THE BASIC STATEMENT .'//)
READ(5,101,END=302)CARD
FORMAT(80A1)
302 WRITE(6,102)
102 FORMAT(20,10X,' A. REM
* , THE BASIC STATEMENT WHICH ALLOWS YOU TO INSERT
* , YOUR PROGRAM IS IDENTIFIED BY THE KEY WORD 'REM'
* , REM IS A NON-EXECUTING STATEMENT WHICH MAY BE USED OPTIONALLY

```

```

LES000020
LES000030
LES000040
LES000050
LES000060
LES000070
LES000080
LES000090
LES000100
LES000110
LES000120
LES000130
LES000140
LES000150
LES000160
LES000170
LES000180
LES000200
LES000210
LES000220
LES000230
LES000240
LES000250
LES000260
LES000270
LES000280
LES000290
LES000300

```





```

303 /* AT ANY PLACE IN YOUR PROGRAM TO INTRODUCE A PROGRAM NAME , TO'//
103 * EXPLAIN VARIABLES, TO DOCUMENT YOUR PROGRAM , ETC.'//)
    READ(5,101,END=303)CARD
    WRITE(6,103)
304 * THE FORM FOR THE REM STATEMENT IS :
104 * REM << ANY SEQUENCE OF ALPHA-NUMERIC CHARACTERS >>
    * THEY REM STATEMENT IS IGNORED BY THE CAIBASIC COMPILER , AND IS'//
    * ONLY FOR YOUR INFORMATION .15X, FOR EXAMPLE :10X, REM PROGR
    * AM TO COMPUTE INCOME TAX'//)
    READ(5,101,END=304)CARD
    WRITE(6,104)
305 * THE PRINT STATEMENT IS THE METHOD OF WRITING OUT THE RESULTS OF'//
105 * THE PRINT STATEMENT , TO DISPLAY VALUES OF VARIABLES , TO LABEL'//
    * THE BASIC PROGRAM , AND TO SKIP A LINE. THE FORM OF THE PRINT'//
    * STATEMENT IS :10X, PRINT << EXPRESSION OR 'STRING' ,>>'//
    * SSION OR 'STRING' ,>>'// MULTIPLE ELEMENTS IN THE PRINT LIST ARE
    * SEPARATED BY COMMAS.'//)
    READ(5,101,END=305)CARD
    WRITE(6,105)
306 * CURRENT VALUE OF 5X, 1. PRINT << EXPRESSION >> WILL WRITE OUT THE'//
106 * IN LESSON 1 WAS A NUMBER, A VARIABLE , WHERE AN EXPRESSION AS DEFINED'//
    * EXPRESSION THAT IS TO BE EVALUATED .10X, FOR EXAMPLE :10X,
    * PRINT 1,A,B1,5*2'// ASSUMING THAT A=10.0 , B=13.3
    * 10.0
    * 13.3
    READ(5,101,END=306)CARD
    WRITE(6,106)
307 * ALPHA-NUMERIC CHARACTERS OF THE STRING WITHIN 'SINGLE' QUOTE
107 * THIS FORM IS USED FOR LABELING THE COMPUTER OUTPUT.
    * FOR EXAMPLE : PRINT 'THE ANSWER IS :',
    * PRODUCES THE RESULT :10X, THE ANSWER IS :10X, 3. PRINT BY
    * ITSELF IS USED TO SKIP A LINE ON THE COMPUTER OUTPUT .'//)
    READ(5,101,END=307)CARD
    WRITE(6,107)
308 * THE COMPUTER OUTPUT SHEET IS DIVIDED INTO 8 ZONES'//
108 * EACH 15 COLUMNS WIDE. PRINT ZONES CAN BE SKIPPED BY PUTTING A'//
    * BLANK IN THE PRINT LIST . FOR EXAMPLE :10X, PRINT A, B, X'//
    * OUTPUTS THE THIRD ZONE , AND PUTS X IN THE FOURTH ZONE .
    * SKIPS THE ALPHA VARIABLES AND 'STRINGS' MAY EXTEND OVER SEVERAL'//
    * ZONES , BUT NUMERIC RESULTS ARE LEFT ADJUSTED IN THE SPECIFIED'//
    * ZONE , IF MORE THAN EIGHT ITEMS OCCUR IN THE PRINT LIST , THE'//
    * ITEMS WILL OVERFLOW AND BE PRINTED ON THE NEXT LINE .'//)
    READ(5,101,END=308)CARD
    WRITE(6,108)
    USING THE PRINT AND END STATEMENT YOU NOW HAVE THE'//
    LES000310
    LES000320
    LES000330
    LES000340
    LES000350
    LES000360
    LES000370
    LES000380
    LES000390
    LES000400
    LES000410
    LES000420
    LES000430
    LES000440
    LES000470
    LES000480
    LES000490
    LES000500
    LES000520
    LES000530
    LES000540
    LES000550
    LES000560
    LES000570
    LES000580
    LES000600
    LES000610
    LES000620
    LES000640
    LES000650
    LES000660
    LES000670
    LES000680
    LES000700
    LES000710
    LES000720
    LES000730
    LES000750
    LES000760
    LES000770
    LES000780

```



```

* FACILITY TO EXECUTE YOUR FIRST PROGRAMS. FOR EXAMPLE : //10X, LES00790
** REM PROGRAM TO COMPUTE THE SQUARE OF A NUMBER //10X, LES00800
** PRINT //5**2//10X, END //10X, LES00810
** PRODUCES THE RESULT : //5X, 5, SQUARED = 25 //10X, LES00820
** YOU WILL NOW BE GIVEN TWO PROBLEMS TO SOLVE. YOU WILL ENTER //10X, LES00830
** THE EXECUTION PHASE OF CAIBASIC WHERE YOU CAN RUN YOUR PROBLEMS. //10X, LES00840
** AND THEN YOU WILL RETURN TO FINISH THE LESSON. //5X, 1) FIND THE //10X, LES00850
** SQUARE ROOT OF 5 SQUARED MINUS 4 TIMES 2 //5X, 2) FIND //10X, LES00860
** THE VALUE OF 3.1416(B CUBED)H/12.// WHERE B=2.50, H=3.03. //10X, LES00880
309 WRITE(6,109) IF YOU WISH TO SKIP THESE PROBLEMS REPLY : YES //10X, LES00880
109 FORMAT(0) LESSON WILL CONTINUE.//)
5 READ(5,101,END=309)CARD
CALL CRUNCH(ILNGTH)
IF(CARDP(1).EQ.ASTRSK)GO TO 5
IF(CARDP(1).EQ.ALPHA(25))GO TO 10
CALL TEST1
310 WRITE(6,110) REPLY WITH THE ANSWER TO THE FIRST PROBLEM.//10X, LES00910
110 READ(5,101,END=310)CARD
CALL CRUNCH(ILNGTH)
IF(CARDP(1).EQ.ASTRSK)GO TO 6
IF(CARDP(1).EQ.DIGIT(4))GO TO 7
WRITE(6,111) YOUR ANSWER IS WRONG. YOUR PRINT STATEMENT SHOULD'//10X, LES00920
111 FORMAT(0) PRINT ((5**2)-(4*2*2))**5'//10X, LES00930
7 WRITE(6,112) REPLY WITH THE ANSWER TO THE SECOND PROBLEM.//10X, LES00940
112 FORMAT(0) REPLY WITH: THE ANSWER TO THE SECOND PROBLEM.//10X, LES00950
8 READ(5,101,END=7)CARD
CALL CRUNCH(ILNGTH)
IF(CARDP(1).NE.DIGIT(3))GO TO 9
IF(CARDP(1).EQ.DIGIT(4))GO TO 10
WRITE(6,113) YOUR ANSWER IS WRONG. YOUR PRINT STATEMENT SHOULD'//10X, LES00960
113 FORMAT(0) PRINT (3.1416*(2.50**3)*3.03)/12 //10X, LES00970
10 WRITE(6,114) READ METHOD WHICH PROVIDES INPUT TO THE //5X, LES00980
114 FORMAT(0) STATEMENT IS : //5X, LES00990
** THE PROGRAM STATEMENT IS : //5X, LES01000
** READ EVERY VARIABLE.//5X, LES01010
** FOR ELEMENT VARIABLE.//5X, LES01020
**// WHEN THE READ STATEMENT INPUT VALUES TO PROGRAM VARIABLES.//2X, LES01030
**// USED TOGETHER IN A DATA STACK. IT TAKES THE TOP ELEMENT OF THE APPROPRIATE DATA.//10X, LES01040
**// SUCCESSIVE READ STATEMENTS FROM A STACK OF ALPHA-NUMERIC DATA. AS EACH VARIABLE LES01050
**E// IS READ, IT TAKES THE TOP ELEMENT OF THE APPROPRIATE DATA.//10X, LES01060
** STACK.//10X, LES01070
**//10X, LES01080
**//10X, LES01090
**//10X, LES01100
**//10X, LES01110
**//10X, LES01120
**//10X, LES01130
**//10X, LES01140
**//10X, LES01150
**//10X, LES01160
**//10X, LES01170
**//10X, LES01180
**//10X, LES01190
**//10X, LES01200
**//10X, LES01210
**//10X, LES01220
**//10X, LES01230
**//10X, LES01240
**//10X, LES01250
**//10X, LES01260

```



```

315 READ(5,101,END=315)CARD
115 WRITE(6,115)
FORMAT(0,10X,' D. DATA
THE DATA STATEMENT IS A LIST OF INPUT NUMBERS OR
// WILL BE ASSIGNED TO VARIABLES IN A READ STATEMENT
// OF THE DATA STATEMENT IS
DATA << NUMBER OR 'STRING' >>
// 5X,
THE 'STRING' OF ALPHA-NUMERIC CHARACTERS MUST BE ENCLOSED
// 5X,
// SINGLE QUOTES
// 5X,
// DATA STATEMENTS MAY BE PLACED ANYWHERE IN A PROGRAM, BUT THERE
// IS AN UPPER LIMIT OF 500 NUMERIC AND 500 ALPHA-NUMERIC DATA
// ELEMENTS FOR EACH PROGRAM
//
READ(5,101,END=316)CARD
316 WRITE(6,116)
FORMAT(0,10X,' WHEN THE FIRST DATA STATEMENT IS INTERPRETED
// BY THE CAIBASIC COMPILER A FIRST IN, FIRST OUT STACK IS FORMED
// FOR THE NUMERIC AND ALPHA-NUMERIC AS EACH NUMBER OR STRING
// IN A DATA LIST IS INTERPRETED, IT IS PLACED ON THE BOTTOM OF
// ITS RESPECTIVE DATA STACK. AS OTHER DATA STATEMENTS ARE LOCATED
// IN THE PROGRAM ITS ELEMENTS ARE PLACED ON THE BOTTOM OF THE
// PROPER STACK. // 10X, ' 1.) EXAMPLE: DATA 10.0,13.33, J.E. SMITH
// 18.75X, ' 2.) DATA Z.X. DOE, 19
// PRODUCES THE FOLLOWING DATA STACKS: // 5X, ' NUMERIC, 10X, ' ALPHA-NUMERIC
// ERIC//5X, ' 10.0, 10X, ' J.E. SMITH//5X, ' 13.33, 10X, ' Z.X. DOE//5X,
// 18.75X, 19 //
317 READ(5,101,END=317)CARD
117 WRITE(6,117)
FORMAT(0,10X,' DURING EXECUTION OF READ STATEMENTS, AS THE
// VARIABLES IN THE READ LIST ARE ASSIGNED VALUES FROM THE TOP OF
// THE APPROPRIATE DATA STACK, THE DATA STACK IS DECREMENTED AND
// THE NEXT ELEMENT THAT THE DATA POPS UP. // 10X, ' 2.) EXAMPLE: READ A,B1,Z
// // ASSUMES IN THE READ LIST ARE ASSIGNED VALUES AS FOLLOWS: // 5X,
// VARIABLES IN THE READ LIST ARE ASSIGNED VALUES AS FOLLOWS: // 5X,
// A RESULTING DATA STACKS ARE AS FOLLOWS: // 5X, ' 19 //
// PHA-NUMERIC//5X, ' 18, 15X, ' Z.X. DOE//5X, ' 19 //
READ(5,101,END=318)CARD
318 WRITE(6,118)
FORMAT(0,10X,' E. RESTORE STATEMENTS ARE USED TO RETURN THE NUMERIC
// THE RESTORE STATEMENTS ARE USED TO RETURN THE NUMERIC
// THAT AND THE ALPHA-NUMERIC DATA STACKS TO THEIR ORIGINAL CONDITION SO
// THAT AND THE ALPHA-NUMERIC DATA MAY BE USED AGAIN. RESTORES NUMERIC DATA
// IS: // 10X, ' ( RESTORES ALPHA-NUMERIC DATA )
// NTORS ( RESTORES ALPHA-NUMERIC DATA )
// YOU WILL NOW BE ASKED SOME QUESTIONS ABOUT THE READ STATEMENTS AS
// RESTORE STATEMENTS: // 10X, ' CONSIDER THE FOLLOWING STATEMENTS AS
// PART OF A BASIC PROGRAM: // 5X, ' DATA 2,5,7,16, 'ANS=
// ,CORRECT, 25, 'WRONG, 0.0//5X, ' READ A,B3,1$C1,D//5X, ' READ J$

```





```

*E4,K$,X'///)
READ(5,101,END=319)CARD
319 WRITE(6,119)
119 FORMAT('O',10X,' 1.) WHAT IS THE VALUE OF C1 ?? REPLY WITH VALUE'//
*)
15 READ(5,101,END=320)CARD
CALL CRUNCH(ILNGTH)
IF(CARDP(1).EQ.ASTRSK) GO TO 15
IF(CARDP(1).EQ.DIGIT(8)) GO TO 20
320 WRITE(6,120)
120 FCRMAT('O',10X,' THE CORRECT ANSWER IS C1=7 • THE DATA IS PUT '//'
* INTO TWO FIFO STACKS, ONE FOR NUMERIC DATA, AND ONE FOR ALPHA-//
** NUMERIC DATA WHEN THE READ STATEMENTS ARE EXECUTED, THE //
** VARIABLES ARE ASSIGNED AS FOLLOWS: //5X, A <-- 2,10X, I$ <-- AN //
*S //5X, B3 <-- 5,10X, J$ <-- CORRECT //5X, C1 <-- 7,10X, K$ <
** -- WRONG //5X, D <-- 16 //5X, E4 <-- 25 //5X, X <-- 0.0 //)
20 WRITE(6,121)
121 FCRMAT('O',10X,' NOW CONSIDER THAT THE FOLLOWING STATEMENTS //
* WERE ADDED TO THE ABOVE PROGRAM: //5X, RESTORE //5X, READ Z4,A //
* 10X, 2.) WHAT IS THE VALUE OF Z4 ?? REPLY WITH VALUE • //)
25 READ(5,101,END=322)CARD
CALL CRUNCH(ILNGTH)
IF(CARDP(1).EQ.ASTRSK) GO TO 25
IF(CARDP(1).EQ.DIGIT(3).AND.CARDP(2).EQ.BLANK) GO TO 30
322 WRITE(6,122)
122 FCRMAT('O',10X,' THE CORRECT ANSWER IS Z4=2 • THE RESTORE COMMAND //
* RETURNS THE NUMERIC DATA STACK TO ITS ORIGINAL CONDITION, AND //
** THE READ STATEMENT ASSIGNS VALUES FROM THE TOP OF THE DATA //
** STACK TO THE VARIABLES IN THE READ LIST AS FOLLOWS: //5X, Z4 <--
* 2 //5X, A <-- 5 //)
30 WRITE(6,123)
123 FCRMAT('O',10X,' F. SUMMARY //
* NOW THAT YOU HAVE SEEN HOW READ AND DATA STATEMENTS WORK TOGETHER, //
** TO INPUT VALUES INTO YOUR PROGRAM, AND HOW THE PRINT STATEMENT //
** IS USED TO OUTPUT AND LABEL RESULTS YOU HAVE THE FACILITY //
** TO WRITE SIMPLE PROGRAMS USING INPUT DATA //5X, FOR EXAMPLE //
** (( 8**2)**.5) COULD BE WRITTEN IN SYMBOLIC FORM AND THE DATA //
** ( A**B)**.5) IN AS FOLLOWS: //5X, READ A,B //5X, PRINT,ANS=, (LES02130
** ( A**B)**.5) //5X, DATA 8,2 //5X, END //) IN THE NEXT LESSON YOU WLES02140
** YOU GREAT FLEXIBILITY IN WRITING EXPRESSIONS //
** AND WILL ALLOW YOU TO DO ASSIGNMENTS SUCH AS: //5X, A=B**2-4*A*C //
** AND Z=(A**2+3)/A, THEN BY SAYING: PRINT A,Z // THE RESULTS OF BLES02170
** OTHER EXPRESSIONS WOULD BE DISPLAYED. //) LLES02180
324 WRITE(6,124)
124 FCRMAT('O',10X,' YOU WILL NOW BE GIVEN SOME REPRESENTATIVE PROBLEMLLES02210
* S'// TO GIVE YOU A CHANCE TO EXERCISE YOUR NEW PROGRAMMING TOOLS //LES02220
LES02230
LES01750
LES01760
LES01770
LES01780
LES01790
LES01800
LES01810
LES01820
LES01830
LES01840
LES01850
LES01860
LES01870
LES01880
LES01890
LES01900
LES01910
LES01920
LES01930
LES01940
LES01950
LES01960
LES01970
LES01990
LES02000
LES02010
LES02020
LES02030
LES02040
LES02050
LES02060
LES02070
LES02080
LES02090
LES02130
LES02140
LES02170
LES02180
LES02190
LES02200
LES02210
LES02220
LES02230

```









```

- - - - - IAPTR, INPTR, IADATA(500), XNDATA(500), STRING(5),
DIGIT(10), IPRTIB(10), LISTST(100), ISTLST(100),
PRT(2500), NERRS, INST, NSTLST, DEBUG, DOLSGN, QUOTE,
EQUALS, PARRT, DECIMAL, PLUS, CMINUS, SLASH, COMMA,
PARLFT, ASTRSK, BLANK
INTERP, IEXERR, /
COMMON LES3/,LESON3
CALL ALLOAD(LS3,N1)
WRITE(6,100)
** THIS INSTRUCTION SET WILL INTRODUCE YOU TO ASSIGNMENT STATEMENTS.//
** AND BUILT-IN FUNCTIONS. THE LET STATEMENT AND THE IO BUILT IN//
** FUNCTION WILL ENABLE YOU TO EVALUATE AND ASSIGN VARIABLES TO //
** COMPLEX ARITHMETIC EXPRESSIONS.///)
READ(5,101,END=302)CARD
FORMAT(6,102)
WRITE(6,102)
FORMAT(10,103), A, LET
** THE LET STATEMENT IS AN ASSIGNMENT OR SUBSTITUTION COMMAND. IT//
** CAUSES THE EVALUATION OF AN EXPRESSION TO BE SUBSTITUTED FOR THE//
** CURRENT VALUE OF A VARIABLE. THE FORM OF THE LET STATEMENT IS://
** 10X, LET <<VARIABLE>> = <<EXPRESSION>> OR //10X,
** LET <<VARIABLE>> = <<VARIABLE>> = ..... = <<EXPRESSION>>///)
READ(5,101,END=303)CARD
FORMAT(6,103)
** THERE MAY BE ANY NUMBER OF VARIABLE = VARIABLE//
** IN THE FORM OF , 10X, , THE LET STATEMENT . AN EXPRESSION IS A NUMBER, //
** VARIABLE, LET OR AN ARITHMETIC EXPRESSION. //10X,
** WHEN THE LET STATEMENT IS EXECUTED THE EXPRESSION ON THE RIGHT//
** SIDE OF THE EQUAL SIGN IS EVALUATED AND THE RESULTING VALUE IS//
** ASSIGNED TO THE VARIABLE OR VARIABLES ON THE LEFT SIDE OF THE EQUA
**//, SIGN. BE LOST. FOR EXAMPLE ://10X, LET A=12.3//10X, LET B
**=14.4//10X, LET A=B=25.//, THE VALUE OF THE VARIABLES A AND B IS
** NOW A=B=25.///)
READ(5,101,END=304)CARD
WRITE(6,104)
** THE ONLY RESTRICTION ON THE USE OF VARIABLES //
** IS THAT SIMPLE AND SUBSCRIPTED VARIABLES CAN ONLY BE ASSIGNED//
** NUMERIC VALUES, AND ALPHA VARIABLES CAN ONLY BE ASSIGNED ALPHA-//
** NUMERIC STRINGS. FOR EXAMPLE ://10X, LET A=D4=X(5)=(3**2 -
**)/4//10X, LET A$=X$, HELP.//10X, LET Y=(B - 4*A*C)**.5//10X.
** LET O$=ANSWER =.///, THE FOLLOWING ASSIGNMENT IS ILLEGAL ://10X
** LET A=D$(3+4).///)
READ(5,101,END=305)CARD
WRITE(6,105)
** YOU WILL NOW BE ASKED QUESTIONS CONCERNING //
** WHAT YOU HAVE JUST LEARNED ::///)

```















```

50 READ(5,101,END=319)CARD
   IF(CARDCP(1)).EQ.A$TRSK) GO TO 50
319 WRITE(6,119)
119 FORMAT(0) NO, THE SQUARE ROOT OF A NEGATIVE NUMBER IS AN
*UNDEFINED OPERATION. IN THE NEXT LESSON YOU WILL BE SHOW A
*BASIC STATEMENT IF THE TEST IS TRUE. FOR EXAMPLE THE ABOVE
*PROGRAM SEQUENCE MIGHT BE ALTERED AS FOLLOWS: //5X, LET B=-9//5X
*, IF B LT 0 THEN 100//5X, 50 LET X=SQR(X)//10X, LET B=ABS(B)//5X
*, 10X, 000//10X, 000//10X, 000//10X, 000//10X, 000//10X
*, GO TO 50//10X, 000//10X, 000//10X, 000//10X, 000//10X
*, FOR A NEGATIVE ARGUMENT, IF TRUE, IT BRANCHES TO STATEMENT NUMB
*ER 100, MAKES THE ARGUMENT POSITIVE, AND BRANCHES BACK TO STATE
*MENT 50, TO COMPLETE THE PROGRAM. //)
READ(5,101,END=320)CARD
320 WRITE(6,123)
123 FORMAT(0) C, SUMMARY
*WITH THE STATEMENTS ( REM, READ, DATA, PRINT ), PLUS THE PREVIOUS
*$// BASIC GAINING AN EFFECTIVE REPERTOIRE FOR PROGRAMMING USE. //
*$FAST GAINING AN EFFECTIVE REPERTOIRE FOR PROGRAMMING USE. //
**THE NEXT LESSON YOU WILL LEARN HOW TO SET UP LOOPS IN A PROGRAM.//
**SO THAT THE MAIN BODY OF A PROGRAM MAY BE EXECUTED AS OFTEN //
**AS DESIRED. AS YOU ARE DOING YOUR REVIEW PROBLEMS, THINK ABOUT//
**HOW YOU COULD SET UP A LOOP TO READ IN ANY AMOUNT OF DATA, //
**PROCESS IT AND THEN HALT FOR SOME TEST CONDITION. //)
READ(5,101,END=324) CARD
324 WRITE(6,124)
124 FORMAT(0) 10X, THE FOLLOWING REVIEW PROBLEMS WILL EXERCISE
*YOUR PROGRAMMING SKILLS TO DATE: //1X, 1.) WRITE A PROGRAM OF YEA
**RS HENCE THE PRESENT FORMULA IS: //10X, P=S( 1/((1+I)*N))// WHERE P
*$ IS THE INTEREST RATE OF A PAYMENT S IN N YEARS HENCE // AT AN
**INTEREST RATE OF PROGRAM TO FIND SIDES A, AND C OF A TRIANGLE C/SIN(C
**2.) WRITE A SINES FORMULA: //10X, A/SIN(A) = B/SIN(B) = C/SIN(C
**// WHERE THE ANGLE REF = (3.1416/180) RADIANS. THE DATA FOR THE PROGR
**SINE OF THE ANGLE, B=34.91 IN, 5X, ANGLE A=98.71 DEG, 5X, ANGLE
**IS: //10X, 1 DEG, 5X, G, 5X, CALL CRUNCH(LENGTH)
**E B=49.97 DEG, 5X, IF(CARDCP(1)).EQ.A$TRSK) GO TO 60
**CALL CRUNCH(LENGTH) IF(CARDCP(1)).EQ. ALPHA(14)) GO TO 65
**IF(CARDCP(1)).EQ. ALPHA(25)) GO TO 70
325 WRITE(6,125)
125 FORMAT(0) YOUR REPLY IS INCORRECT, REPLY AGAIN. //)
65 WRITE(6,126)

```



```

126 FORMAT('0', ' THE ANSWERS TO THE PROBLEMS WILL NOW BE GIVEN SO '//
* THAT YOU MAY CHECK YOUR RESULTS LATER : '//5X, '1.') $1072.74/5X, ' 2
* ) SIDE A=45.06 IN. AND SIDE C=23.69 IN. '//)
CALL EXIT
70 CALL TEST1
WRITE(6,127)
127 FORMAT('0', '10X, ' REPLY WITH ANSWER TO QUESTION 1.) ??'//) .
75 READ(5,101,END=80) CARD
CALL CRUNCH(ILNGTH)
IF(CARDP(1).EQ.ASTRSK) GO TO 75
IF(CARDP(1).NE.DIGIT(2).OR.CARDP(4).NE.DIGIT(3)) GO TO 80
IF(CARDP(1).EQ.DECMAL.AND.CARDP(6).EQ.DIGIT(8)) GO TO 90
80 WRITE(6,128)
128 FORMAT('0', '10X, ' THE CORRECT ANSWER IS $1075.74 AND THE PROGRAM '//
* SHOULD HAVE BEEN SIMILAR TO : '//5X, ' READ S,I,N//5X, ' LET P=S*(1/
* ((1+I)*N))//5X, ' PRINT//PRESENT WORTH= ',P//5X, ' DATA 5000,.08,20,
* /5X, ' END'//)
90 WRITE(6,129)
129 FORMAT('0', '10X, ' REPLY WITH ANSWER TO SIDE A FOR QUESTION 2.)'//)
91 READ(5,101,END=95) CARD
IF(CARDP(1).EQ.ASTRSK) GO TO 91
IF(CARDP(1).NE.DIGIT(5).OR.CARDP(2).NE.DIGIT(6)) GO TO 95
IF(CARDP(3).EQ.DECMAL.AND.CARDP(4).EQ.DIGIT(1)) GO TO 97
95 WRITE(6,130)
130 FORMAT('0', '10X, ' THE CORRECT ANSWER IS SIDE A=45.06 IN. AND
* SIDE C=23.69 IN. ' THE PROGRAM SHOULD HAVE BEEN SIMILAR TO : '//5X,
* READ A,B,C,B1//5X, ' LET A1=(B1*SIN(A*(3.1416/180)))/SIN(B*3.141
* 6/180) )//5X, ' LET C1=(B1*SIN(C*(3.1416/180)))/SIN(B*(3.1416/180)) ,
* /5X, ' PRINT//SIDE A= ',A1, 'SIDE C = ',C1//5X, ' DATA 98.71,49.97,
* 31.32,34.91//5X, ' END'//)
97 CALL EXIT
END

```

CCCC

THIS LESSON INTRODUCES BRANCHING , BOTH CONDITIONAL AND UNCONDITIONAL .

```

COMMON
STACK(100), PROG(200), CARD(80), CARDP(80), CARDP(8C), ALPHA(48),
IAPTR, INPTR, IADATA(500), XNDATA(500), STRING(5),
DIGIT(10), IPRITB(10), LISTST(100), ISTLST(100),
PRT(2500), NERRS, INST, NSTLST, DEBUG, DOLSGN, QUOTE,
EQUALS, PARRT, PLUS, CMINUS, SLASH, COMMA,
PARLFT, ASTRSK, BLANK
COMMON INTRP, IEXERR
REAL*8 LES4/LES4,N1
CALL ALOAD(LES4,N1)
WRITE(6,100)
100 FCFORMAT('0', '5X, ' *** LESSON 4 ***
* * * * *
LES000030
LES000040
LES000050
LES000060
LES000070
LES000080
LES000090
LES000100
LES000110
LES000120
* * *

```









```

*VALUE OF THE EXPRESSION , FOR EXAMPLE : '//5X,' LET I=3'/5X,' ON I
*GO TO 100,33,475,9999://)
READ(5,101,END=306)CARD
306 WRITE(6,106)
106 FORMAT(0), EXECUTION OF THE 'COMPUTED GO TO' WOULD CAUSE '/'
*PROGRAM CONTROL TO TRANSFER UNCONDITIONALLY TO STATEMENT NUMBER 47
*5 '/5X,' YOU MUST BE CAREFUL WHEN USING THE 'COMPUTED GO TO'//
** NOT ONLY BECAUSE OF INFINITE LOOPS ; BUT BECAUSE THE EXPRESSION//
** MUST BE AN INTEGER BETWEEN 1-->9999 , AND THERE MUST BE A '/'
** STATEMENT NUMBER FOR 'ALL' POSSIBLE VALUES OF THE EXPRESSION //
** )
READ(5,101,END=307)CARD
307 WRITE(6,107)
107 FORMAT(0),5X,' B. CONDITIONAL BRANCHING
*THE CONDITIONS ARE TRUE . IF THE TEST OF RELATIONS IS TRUE THEN '/'
**RELATIONS ARE TRUE ; OTHERWISE PROGRAM CONTROL CONTINUES//
**TRANSFER OF CONTROL OCCURS ; THE FORM OF THE CONDITIONAL BRANCH IS:
* '// 1X,' IF << EXPRESSION >> << EXPRESSION >> THEN
** << STATEMENT NUMBER >> //, NOTE THAT ALPHA VARIABLES ARE NOT ALLOWED
** AS AN EXPRESSION IN A '/' CONDITIONAL BRANCH . '/' )
READ(5,101,END=308)CARD
308 WRITE(6,108)
108 FORMAT(0),5X,' THE RELATIONS ARE :
*SYMBOLS ,2X,' EXAMPLE ,5X,' MEANING
** '// 2X,' GT ,5X,' A GREATER THAN B'/2X,' GE ,5X,' A GE
** B ,5X,' LT ,5X,' A LESS THAN B'/2X,' LE ,5X,' A LE
** A LESS THAN B'/2X,' NE ,5X,' A NOT EQUAL TO B'/3X,' = ,5X,' A =
** B'/2X,' EQ ,5X,' A EQUAL TO B'//)
READ(5,101,END=309)CARD
309 WRITE(6,109)
109 FORMAT(0),5X,' WHEN THE CONDITIONAL STATEMENT IS EXECUTED THE '/'
* ( EXPRESSION ) IS TESTED , AND IF THE RELATION//
** IS TRUE , THEN CONTROL IS TRANSFERRED TO THE STATEMENT NUMBER.
** OTHERWISE , PROGRAM CONTROL CONTINUES TO THE NEXT SEQUENTIAL '/'
** STATEMENT . FOR EXAMPLE : '//5X,' I READ A'/5X,' //5X,'/5X,
** IF A LT 0 THEN 99//5X,' LET X=SQR(A)'/5X,' //5X,'/5X,
** PRINT ILLEGAL ARGUMENT , A'/5X,' GO TO 1//5X,'// THE ONLY
** TIME THAT THE CONDITIONAL BRANCH IS EXECUTED IS WHEN '/' A IS
** HAN 0 . '/' )
READ(5,101,END=310)CARD
310 WRITE(6,110)
110 FORMAT(0), YOU WILL NOW BE ASKED SOME QUESTIONS ABOUT BRANCHING
* '// 10X,' 1. ) ARE THE FOLLOWING STATEMENTS CORRECT ,/ REPLY: YES OR
** NO) //15X,' GO TO END//)
5 READ(5,101,END=311)CARD
CALL CRUNCH(LENGTH)
LES00610
LES00620
LES00630
LES00640
LES00650
LES00660
LES00670
LES00680
LES00690
LES00700
LES00710
LES00720
LES00730
LES00740
LES00750
LES00760
LES00770
LES00780
LES00820
LES00830
LES00840
LES00850
LES00860
LES00870
LES00880
LES00890
LES00900
LES00910
LES00920
LES00930
LES00940
LES00950
LES00960
LES00970
LES00980
LES00990
LES01000
LES01010
LES01020
LES01030
LES01040
LES01070
LES01080

```





```

IF(CARDP(1).EQ.ASTRSK) GO TO 5
IF(CARDP(1).EQ.ALPHA(14))GO TO 10
311 WRITE(6,111) YOUR ANSWER IS INCORRECT • THE ONLY THING THAT IS •/
111 *ALLOWED AFTER A •GO TO• IS A STATEMENT NUMBER BETWEEN 1-->9999•/
*//)
10 WRITE(6,112)
112 FORMAT(•0•,15X,• IF X1 LT A$ THEN 10•//)
15 READ(5,101,END=313)CARD
CALL CRUNCH(ILNGTH)
IF(CARDP(1).EQ.ASTRSK) GO TO 15
IF(CARDP(1).EQ.ALPHA(14)) GO TO 20
313 WRITE(6,113)
113 *BE USE FOR TESTING PURPOSES IN A CONDITIONAL STATEMENT MAY NOT•/
20 WRITE(6,114)
114 FORMAT(•0•,15X,• IF(X**3-4) GE 10 THEN GO TO 100•//)
25 READ(5,101,END=315)CARD
CALL CRUNCH(ILNGTH)
IF(CARDP(1).EQ.ASTRSK) GO TO 25
IF(CARDP(1).EQ.ALPHA(14)) GO TO 30
315 WRITE(6,115)
115 *AFTER THEN IS A STATEMENT NUMBER •//)
30 WRITE(6,116)
116 FORMAT(•0•,15X,• LET Y=6,•/15X,• ON Y GO TO 1,3,5,7,999•//)
35 READ(5,101,END=317)CARD
CALL CRUNCH(ILNGTH)
IF(CARDP(1).EQ.ASTRSK) GO TO 35
IF(CARDP(1).EQ.ALPHA(14)) GO TO 40
317 WRITE(6,117)
117 *STATEMENT NUMBERS IN THE • COMPUTED GO TO••//)
40 WRITE(6,118)
118 *EPLY: YES OR NO)•//5X,• 1 READ A,B•/5X,• 3 LET Z=A*B•/5X,• IF Z L
*10 GO TO 999•/5X,• •••/5X,• •••/5X,• GO TO 3,5,7,
*10/5X,• 999 END•//)
READ(5,101,END=319)CARD
319 WRITE(6,119)
119 *COULD BE CORRECTED BY CHANGING THE GO TO 3, TO GO TO 1,
** WHEN YOU WRITE PROGRAMS HAVING BRANCHES CR LOOPS YOU MUST ALWAYS,
**/ •CONSIDER HOW THE PROGRAM WILL STOP. YOU HAVE OBSERVED THAT •/
**AN •INFINITE LOOP• CAN BE STOPPED BY HAVING A READ STATEMENT •/
**IN THE LOOP AND JUST RUN OUT OF DATA • HOWEVER, ENDING A PROGRAM,
**/ ON AN ERROR IS AN INELIGANT METHOD • THE MOST COMMON METHODS•//
** USE THE •GO TO• AND •IF/THEN• COMMANDS TO CONTROL PROGRAM•//

```



















```

**AND THAT THE NEXT IS THE SAME AS THE SIMPLE VARIABLE FOLLOWING. FOR. //
**BECAUSE THE INCREMENT AND VALUE OF THE LOOP. //5X,
**STEP. MODIFIER AND ITS EXPRESSION MAY BE OMITTED, AND THE. //
**INCREMENT WILL BE ASSUMED TO BE ONE(+1). FOR EXAMPLE THE. //
**ABOVE STATEMENT COULD BE WRITTEN: AFTER STEP. MAY BE. //5X,
**FOR I=1 TO 10.//5X, THE INCREMENT VALUE AFTER STEP. MAY BE. //5X,
**POSITIVE OR NEGATIVE ALLOWING THE FLEXIBILITY OF LOOPING FORWARD. //
**OR BACKWARD. FOR A NEGATIVE STEP, VALUE THE TEST BECOMES. //
**LESS THAN. FOR EXAMPLE THE FOLLOWING. FOR STATEMENTS ARE. //
**EQUIVALENT. //5X, FOR I=1 TO 10.//5X, FOR I=10 TO 1 STEP -1.//
**READ(5,101,END=305)CARD
305 WRITE(6,105)
135 FORMAT(6,105)
**// ANOTHER USEFUL TECHNIQUE OF LOOPING IS "NESTING".
**THE INNER LOOP SPINS AROUND AS MANY TIMES AS THE OUTER LOOP. //
**IS INCREMENTED. FOR EXAMPLE CONSIDER THIS PROGRAM. //5X,
**FOR I=1 TO 10.//5X, FOR J=1 TO 20 STEP 2.//6X, //6X, //5X,
**NEXT J.// IN THIS EXAMPLE THE "OUTSIDE LOOP(I)" IS. //
**REPEATED 10 TIMES, AND THE "INNER LOOP(J)" WOULD BE REPEATED 20. //
**// LOOPS MAY BE NESTED UP TO A MAXIMUM OF 20; HOWEVER, THEY. //
**CANNOT OVERLAP. THE INNERMOST LOOP MUST BE CLOSED WITH. //
**NEXT. STATEMENT BEFORE ENCOUNTERING THE NEXT OUTER LOOP. //5X,
**NEXT X=10 TO 1 STEP -1.//5X, FOR Y=3 TO 5.//5X, FOR Z=-3 TO -5 STEP
** -1.//6X, //6X, //5X, NEXT Z.//5X, NEXT Y.//5X, NEXT X.//
**READ(5,101,END=306)CARD
306 WRITE(6,106)
106 FORMAT(6,106)
**// WITHIN A FOR/NEXT LOOP CONDITIONAL AND UNCONDITIONAL
**OR WITHIN BRANCHES OF THE SAME LOOP, HOWEVER, IT IS NOT POSSIBLE. //
**TO BRANCH INTO THE MIDDLE OF A FOR/NEXT LOOP BECAUSE LOGIC. //
**PROBLEMS OCCUR AND AN ERROR WILL RESULT. AN ADDITIONAL ITEM. //
**TO BE CAREFUL ABOUT IS USING THE INDEX VARIABLE OF THE LOOP. INDEX. //
**LOOP IN COMPUTATIONS. IF YOU ALTER THE VALUE OF THE LOOP. //5X,
**FOR I=1 TO 10.//5X, LET I=I+1.//5X, PRINT I.//5X, EN
**D.//. REPLY WITH VALUE OF I THAT IS PRINTED. //
**READ(5,101,END=307)CARD
5 CALL CRUNCH(1,LENGTH)
**IF(CARDP(1).EQ.ASTRSK) GO TO 5
**IF(CARDP(1).EQ.DIGIT(2).AND.CARDP(2).EQ.DIGIT(2)) GO TO 10
307 WRITE(6,107)
107 FORMAT(6,107)
**YOUR ANSWER IS INCORRECT. THE LOOP INDEX(I) IS. //
**ALTERED IN THE LOOP (I=11), AND THEN INCREMENTED AND TESTED (I=12).
**// THE FINAL VALUE OF I=12.//
10 WRITE(6,108)

```



```

108 FORMAT('O', ' YOU WILL NOW BE ASKED SOME QUESTIONS ABOUT WHAT YOU HAVE JUST LEARNED ',//)
109 WRITE(6,109)
110 * * * * * 1.) SAMPLE PROGRAM, /15X, ' LET S=0, /15X, ' FOR K=-5
111 * * * * * LET S=S+K, /15X, ' NEXT K, /15X, ' PRINT, SUM= ', S, /15X, '
112 * * * * * END, //, ' REPLY WITH VALUE OF S ', //)
113 * * * * * READ(5,101,END=310) CARD
114 * * * * * CALL CRUNCH(1,LENGTH)
115 * * * * * IF(CARDP(1).EQ.ASTRSK) GO TO 15
116 * * * * * IF(CARDP(1).EQ.DIGIT(1).AND.CARDP(2).EQ.BLANK) GO TO 20
117 * * * * * WRITE(6,110)
118 * * * * * IN THE LOOP ARE: //5X, ' K, 5X, ' S, /5X, ' -5, /5X, ' -4, /5X, ' -9
119 * * * * * //5X, ' -3, /5X, ' -2, /5X, ' -1, /5X, ' -15, /5X, ' 0, /5X, ' -
120 * * * * * //5X, ' 1, /5X, ' -14, /5X, ' 2, /5X, ' -12, /5X, ' 3, /5X, ' -9, /5X, ' 4, /5X, '
121 * * * * * //5X, ' 5, /5X, ' 6, //, ' S=0, //)
122 * * * * * WRITE(6,111)
123 * * * * * 2.) SAMPLE PROGRAM, /15X, ' FOR I=100 TO 1 STEP 2
124 * * * * * //15X, ' LET Z=I**2, /15X, ' LET Y=Z+10, /15X, ' NEXT I, /15X, ' PRINT I,
125 * * * * * //15X, ' END, //, ' REPLY WITH VALUE OF I, //)
126 * * * * * READ(5,101,END=312) CARD
127 * * * * * CALL CRUNCH(1,LENGTH)
128 * * * * * IF(CARDP(1).EQ.ASTRSK) GO TO 25
129 * * * * * IF(CARDP(1).EQ.DIGIT(2).AND.CARDP(3).EQ.DIGIT(1)) GO TO 30
130 * * * * * WRITE(6,112)
131 * * * * * YOUR ANSWER IS INCORRECT. THE INDEX VALUE OF THE, //
132 * * * * * LOOP(I=100) IS GREATER THAN THE FINAL VALUE(1) IN THE FIRST, //
133 * * * * * *TEST. * * * * * IF THE LOOP WERE DECREMENTED IN STEPS OF -2, //,
134 * * * * * * THEN THE LOOP WOULD BE EXECUTED AND THEN I=0. //)
135 * * * * * WRITE(6,113)
136 * * * * * 3.) SAMPLE PROGRAM, /15X, ' LET T=C, /15X, ' FOR I=5
137 * * * * * * TO 1 STEP -1, /15X, ' 10, /15X, ' IF I=J THEN 10, /15X, ' GO
138 * * * * * * TO 15, /15X, ' 10, LET I=T+1, /15X, ' PRINT I, /15X, ' 15 NEXT J, /15X, '
139 * * * * * * NEXT I, /15X, ' END, //, ' REPLY WITH VALUE OF T ', //)
140 * * * * * READ(5,101,END=314) CARD
141 * * * * * CALL CRUNCH(1,LENGTH)
142 * * * * * IF(CARDP(1).EQ.ASTRSK) GO TO 35
143 * * * * * IF(CARDP(1).EQ.DIGIT(6).AND.CARDP(2).EQ.BLANK) GO TO 40
144 * * * * * WRITE(6,114)
145 * * * * * YOUR ANSWER IS INCORRECT. THE OUTSIDE LOOP(I)
146 * * * * * IS REPEATED 5 TIMES (I=5-->1); AND FOR EACH VALUE OF I, THE
147 * * * * * * INNER LOOP(J) IS REPEATED 5 TIMES (J=1-->5). THEREFORE, //,
148 * * * * * * THE CONDITIONAL BRANCH IS EXECUTED FIVE TIMES: //5X, ' I=5, ' J=5, /5X,
149 * * * * * * *X, ' I=4, ' J=4, /5X, ' I=3, ' J=3, /5X, ' I=2, ' J=2
150 * * * * * * *X, ' I=1, ' J=1, //, ' THUS T=5, //)
151 * * * * * WRITE(6,115)
152 * * * * * 4.) SUBSCRIPTED VARIABLES/LISTS AND TABLES, /5X, '
153 * * * * * 1. IN LESSON 1 YOU LEARNED THAT SIMPLE VARIABLES WERE A LETTER, //,

```



```

**OR A LETTER FOLLOWED BY A DIGIT. SUBSCRIPTED VARIABLES CONSIST OF A LETTER FOLLOWED BY A SINGLE OR DOUBLE SUPSCRIPT IN PARENTHESES.
**IS. THE SUBSCRIPTS MAY BE ANY LEGAL EXPRESSION THAT EVALUATES TO AN INTEGER VALUE. FOR EXAMPLE:
**A(1), B(3), D(1,3), Z(I,J), X(3#A), D(A(1)) ARE LEGAL SUBSCRIPTED VARIABLES. BUT A(1), Z(1,2,3) ARE ILLEGAL SUBSCRIPTED VARIABLES. A USE OF SUBSCRIPTED VARIABLES FOLLOWS:
READ(5,101,END=316)CARD
316 WRITE(6,116)
116 FORMAT(0,5X, ' 2. A NUMERIC LIST, OR VECTOR, OR SINGLE DIMENSIONED ARRAY, IS A SET OF NUMERIC VALUES ARRANGED IN AN ORDERLY MANNER. FOR EXAMPLE, SUPPOSE YOU WOULD LIKE TO READ SCME NUMBERS INTO YOUR PROGRAM AND HAVE THESE NUMBERS AVAILABLE AND IDENTIFIED FOR USE AT A LATER TIME. YOU COULD ASSIGN SIMPLE VARIABLES TO EACH VALUE, BUT WHAT IF YOU HAD 100 NUMBERS? IT IS EASIER TO THINK OF THE NUMBERS AS A LIST OF VALUES OR A VECTOR THE SIZE OF WHICH IS DETERMINED BY HOW MANY NUMBERS YOU HAVE. YOU THEN SIMPLY ASSIGN THE VALUES TO THE LIST.
READ(5,101,END=317)CARD
317 WRITE(6,117)
117 FORMAT(0, ' THE FOLLOWING PROGRAM WILL ASSIGN 10 VALUES TO THE LIST, WHICH CONTAINS 10 ELEMENTS:
L=1 TO 10/5X, ' READ A(L)/5X, ' DATA 1,3,7,5,6,9,4,2,
14,4/5X, ' END// ' THE LIST A, CONTAINING 10 ELEMENTS, IS REPRESENTED BY THE SUBSCRIPTED VARIABLE A(L). READ A(L) IN A FOR/NEXT LOOP, AND WITHIN THE LOOP THE LIST ELEMENTS A(1)-->A(10) LESO1670
ARE ASSIGNED VALUES. NOW THE LIST A HOLDS THE 10 VALUES AND EALLESO1680
EACH VALUE IS IDENTIFIABLE. CONSIDER THE FOLLOWING PROBLEM WHICHLESO1700
SEARCHES A LIST TO FIND THE LARGEST ELEMENT:
READ(5,101,END=318)CARD
318 WRITE(6,118)
118 FORMAT(0,5X, ' DIM N(10)/5X, ' FOR I=1 TO 10/5X, ' LET N(I)=0/5XLESO1740
' NEXT I/5X, ' READ K/5X, ' FOR J=1 TO K/5X, ' READ N(J)/5X,LESO1750
' NEXT J/5X, ' LET L=N(J)/5X, ' IF N(J) LT L THENLESO1760
' L=N(J)/5X, ' LET L=N(J)/5X, ' 10 NEXT J/5X, ' PRINT 'LARGEST NUMBER=
' L/5X, ' DATA 5,3,5,9,10,6/5X, ' END// ' REPLY WITH VALUE OF L//
)
45 READ(5,101,END=319)CARD
CALL CRUNCH(ILNGTH)
IF(CARDP(1).EQ.ASTRSK) GO TO 45
IF(CARDP(1).EQ.DIGIT(2).EQ.DIGIT(1)) GO TO 50
319 WRITE(6,119)
119 FORMAT(0, ' YOUR ANSWER IS INCORRECT. L IS ASSIGNED VALUES AS FOLLESO1850
LLOWS: /5X, ' L=3-->9//)
50 WRITE(6,120)
120 FORMAT(0,5X, ' THE FIRST FOR/NEXT LOOP ZEROS OUT THE LIST. THE FIRST FOR/NEXT LOOP PUTS SOME VALUES IN THE LISTS YOU USE. IT IS A GOOD PRACTICE TO PUT SOME VALUES IN THE LISTS YOU USE. OTHERWISE THE COMPUTER MAY ASSIGN RANDOM VALUES. BY PUTTING

```











```

ELEMENTS OF THE TABLE ON THE MAJOR DIAGONAL (UPPER LEFT TO LOWER
*RIGHT),, THE SUM=15.,,/)
70 WRITE(6,127)
127 FORMAT(0,5X, ' YOU HAVE SEEN HOW SUBSCRIPTED VARIABLES ARE ',,
*USED TO SET UP LISTS(VECTORS) AND TABLES(MATRICES). HOWEVER TO',,
*USE LISTS AND TABLES IN A PROGRAM, YOU MUST DIMENSION THE',,
*MAXIMUM SIZE OF YOUR LIST OR TABLE, SO THAT ENOUGH SPACE WILL',,
*BE ALLOCATED IN THE COMPUTER MEMORY. THIS DIMENSIONING IS DONE',,
*WITH THE DIM STATEMENT.,,/)
READ(5,101,END=328)CARD
328 WRITE(6,128)
128 FORMAT(0,10X, ' C. DIM STATEMENT TELLS THE COMPUTER THE MAXIMUM SIZE OF VECTORS',,
*THE DIM STATES THAT WILL BE USED IN YOUR PROGRAM. THE DIM STATEMENT',,
*AND TABLES THAT APPEAR IN THE PROGRAM BEFORE ANY REFERENCE IS MADE',,
*,, MUST APPEAR IN THE PROGRAM. IN GENERAL PRACTICE THE DIM STATEMENT',,
*TO THE LIST OR TABLE IN STATEMENT IN THE PROGRAM. THE FORM OF THE',,
*IS USUALLY THE FIRST STATEMENT IN THE PROGRAM. DIM <<LIST VARIABLE>> ( <<SIZE>> ),,/)
**DIM STATEMENT IS ://10X, ' DIM <<LIST VARIABLE>> ( <<SIZE>> ),,/)
10X, ' DIM <<TABLE VARIABLE>> ( <<SIZE>> ),,/)
READ(5,101,END=329)CARD
329 WRITE(6,129)
129 FORMAT(0,10X, ' THE LIST AND TABLE VARIABLES ARE SUBSCRIPTED VARIABLE
*S.,, AS DEFINED EARLIER. THE SIZE IS AN UNSIGNED INTEGER IN',,
*PARENTHESES WHICH DENOTES THE MAXIMUM SIZE OF THE LIST OR TABLE',,
*THE DIM STATES MAY CONTAIN A NUMBER OF LISTS OR TABLES',,
*WITH THEIR SIZES SEPARATED BY COMMAS, FOR EXAMPLE :',,
**DIM A(6),X(10,3),Z(14,21),,/)
READ(5,101,END=330)CARD
330 WRITE(6,130)
130 FORMAT(0,5X, ' IF YOU TRY TO REFERENCE AN ELEMENT IN A LIST',,
*FOR TABLE BEYOND THE MAXIMUM SIZE IN THE DIM STATEMENT YOU WILL',,
*GET AN ERROR. THE MAXIMUM SIZE LIST(VECTOR) OR TABLE(MATRIX) ',,
*ALLOWED BY THE CAI-BASIC COMPILER IN ANY ONE PROGRAM IS A TOTAL',,
*OF 1600 COMPILES. QUESTIONS :',,
**YOU WILL THE FOLLOWING DIM STATEMENTS CORRECT, REPLY : YES OR NO',,/)
10X, ' DIM A(10),,/)
75 READ(5,101,END=331)CARD
CALL CRUNCH(ILNGTH)
IF(CARDP(1).EQ.ASTRK) GO TO 75
IF(CARDP(1).EQ.ALPHA(14)) GO TO 80
331 WRITE(6,131)
131 FORMAT(0,10X, ' THE STATEMENT IS INCORRECT. SUBSCRIPTED VARIABLES',,
*ARE A SINGLE',,
*PARENTHESES',,/)
80 WRITE(6,132)
132 FORMAT(0,10X, ' DIM X(500),B(1,10),C(5,5,5),,/)
85 READ(5,101,END=333)CARD

```



```

CALL CRUNCH(ILNGTH)
IF(CARDP(1).EQ.ASTRSK) GO TO 85
IF(CARDP(1).EQ.ALPHA(14)) GO TO 90
333 FORMAT(6,133) THE STATEMENT IS INCORRECT. C(5,5,5) IS ILLEGAL.
133 *ONLY SINGLE OR DOUBLE SUBSCRIPTS ARE ALLOWED.
90 FORMAT(6,134)
134 *IN THIS LESSON, YOU HAVE LEARNED HOW TO USE THE FOR/NEXT STATEMENT.
**//. AND HOW TO MANIPULATE LISTS(VECTORS) AND TABLES(MATRICES).
**BY USING SUBSCRIPTS AND THE DIMENSION STATEMENT. YOU NOW
**HAVE ALL THE TOOLS TO BEGIN WRITING PROGRAMS.
**AND AS YOU WRITE MORE COMPLICATED PROGRAMS, YOU WILL FIND A
**NEED FOR SUBROUTINES. SUBROUTINES ARE COMMONLY USED IN PROGRAMS.
**SEGMENTS THAT ARE USED OVER AGAIN IN OTHER PARTS OF YOUR PROGRAM.
**SUBROUTINES ALLOW YOU TO BRANCH IN TO THE COMMONLY USED SEGMENT.
**AND THEN RETURN TO WHERE YOU WERE AND CONTINUE EXECUTING.
**SUBROUTINE CAN BE CALLED FROM ANYWHERE IN YOUR PROGRAM.
**GOSUB STATEMENT ALLOWS SUBROUTINES IN BASIC, AND YOU WILL BE
**INTRODUCED TO IT IN THE NEXT LESSON.
READ(5,101,END=335)CARD
335 WRITE(6,135)
135 *YOU WILL NOW BE GIVEN TWO OPTIONAL PROGRAMMING.
**PROBLEMS TO THE NUMBERS. //5X. 1.) WRITE A PROGRAM
**O REVERSE THE NUMBERS IN A 10-ELEMENT LIST. X...//. IN OTHER WORDS
**INTERCHANGE X(1) WITH X(10), X(2) WITH X(9), ETC. // READ IN TEN
**ALUES. //5X, 2.) YOUR PROGRAM BY PRINTING THE LIST. // BEFORE AND AFTER
**ER. //5X, 3.) WRITE A PROGRAM TO ARRANGE THE FOLLOWING LIST IN
**//. DECENDING ORDER: 10,30,5,15,40. ONE METHOD OF APPROACHING
**HIS. // IS TO CHECK THE FIRST ELEMENT OF THE LIST AGAINST THE SECOND
**ED. // IF THE FIRST IS NOT LARGER THEN EXCHANGE THE TWO. OTHERWISE
**ATED. // GO AND COMPARE THE NEXT TWO IN THE LIST. THIS PROCESS IS REPE
**//. UNTIL THERE ARE NO MORE EXCHANGES TO BE MADE. A COUNT OF
**THE EXCHANGES CAN BE MADE, AND WHEN THE COUNT EQUALS O THE LI
ST IS IN ORDER.
READ(5,101,END=336)CARD
336 WRITE(6,136)
136 *IF YOU WANT TO EXECUTE THESE PROBLEMS REPLY : YES ,OT
95 *PERWISE NO.
**//.
READ(5,101,END=337)CARD
95 *CALL CRUNCH(ILNGTH)
IF(CARDP(1).EQ.ASTRSK) GO TO 95
337 WRITE(6,137)
137 *S REPLY : YES, OTHERWISE NO.
96 *READ(5,101,END=338)
CALL CRUNCH(ILNGTH)

```



```

IF(CARDP(1).EQ.ASTRSK) GO TO 96
IF(CARDP(1).EQ.ALPHA(14)) CALL EXIT
338 FORMAT(6,1,38)
138 * A LIST.//5X, REM PROBLEM 1.//5X, REM PROGRAM TO REVERSE ELEMENTS IN
* SWAPPED.//5X, REM Y IS A TEMPORARY LOCATION FOR THE ELEMENT BEING
* LET X(1)=X(11-1).//5X, LET X(11-1)=Y.//5X, NEXT I.//5X, REM S.//5X, P
* ROBLEM 2.//5X, REM SORT LIST IN DECREASING ORDER.//5X, REM S.//5X, P
* THE EXCHANGE COUNTER, Y- IS A TEMPORARY LOCATION.//5X, DIM D(5)
* //5X, FOR I=1 TO 5.//5X, READ D(I).//5X, 5 LET S=C.//5X
* ,//5X, FOR I=1 TO N-1.//6X, IF D(I) GE D(I+1), THEN 16.//5X, LET Y=D(I)
* ,//6X, LET D(I)=D(I+1).//6X, LET D(I+1)=Y.//6X, LET S=1.//5X, 16 NEXT
* I.//5X, IF S NE 0 THEN 5.//5X, PRINT NUMBERS IN ORDER.//5X, FOR
* I=1 TO 5.//5X, PRINT D(I).//5X, NEXT I.//5X, END.//
CALL EXIT
END
LES03340
LES03360
LES03370
LES03380
LES03390
LES03410
LES03420
LES03440
LES03460
LES03470
LES03490

```

# THIS LESSON INTRODUCES SUBROUTINES AND RECURSION

```

COMMON
STACK(100), PROG(2000), CARD(80), CARDP(80), ALPHA(48),
IAPTR, INPTR, IADATA(500), XNDATA(500), STRING(5),
DIGIT(10), IPRIB(10), LISTST(100), ISTLST(100),
PRT(2500), NERRS, INST, NSTLST, DEBUG, DOLSGN, QUOTE,
EQUALS, PARRT, DECMAL, PLUS, CMINUS, SLASH, COMMA,
PARLFT, ASTRSK, BLANK
PARLFT, ASTRSK, BLANK
INTERP, IEXERR
REAL*8 LES6, LESON6
CALL ALOAD(LES6, N1)
WRITE(6,100)
100 * THIS LESSON 6 *** LESSON 6 YOU TO SUBROUTINES AND THE PROGRAMMING.
* // TECHNIQUE CALLED RECURSION.//5X, A. SUBROUTINES //5X,
* IN MANY PROGRAMS A BLOCK OF STATEMENTS MAY BE NEEDED ON MORE
* THAN ONE OCCASION. A TECHNIQUE TO REPEATITION OF A BLOCK OF STATEMENTS.
* IS CALLED A SUBROUTINE. SINCE THE REPEITION OF A BLOCK OF INSTRUCTIONS.
* IS ://5X, GOSUB << STATEMENT NUMBER >> //5X, ... //5X,
* READ(5,101, END=302) CARD
FORMAT(6,102)
302 WRITE(6,102)
102 * COMPUTER CONTROL TO THE GOSUB STATEMENT CAUSES THE GOSUB.//
* // WHEN CONTROL IS TRANSFERRED TO THE STATEMENT NUMBER AFTER THE GOSUB.//
* IALLY.// UNTIL A RETURN.// IS ENCOUNTERED. AT THAT TIME, CONTROL
* // IS RETURNED TO THE NEXT STATEMENT FOLLOWING THE GOSUB.//
* STATEMENT WHICH IS CALLED. THE SUBROUTINE. FOR EXAMPLE CONSIDER.//

```

```

LES00020
LES00030
LES00040
LES00050
LES00060
LES00070
LES00080
LES00090
LES00100
LES00110
LES00120
LES00140
LES00150
LES00160
LES00170
LES00180
LES00190
LES00200
LES00210
LES00220
LES00230
LES00240
LES00250
LES00260
LES00270
LES00280
LES00290

```





```

* THE PROGRAM SEGMENT : '///'
  303 READ(5,101,END=303)CARD
  103 WRITE(6,103)
    * B,C,5X, '.../5X, ' PRINT A,B,C,D,E,/5X, ' GOSUB 500,/5X, ' FO
    * R I=1 TO E,/5X, ' .../5X, ' .../5X, ' 500 REM SUBROUTINE,
    * /5X, ' LET X=A+B,/5X, ' LET Z=B+C,/5X, ' LET R=SQR(D),/5X, ' RETURN,/5
    * X, '...///'
  304 READ(5,101,END=304)CARD
  104 WRITE(6,104)
    * B,C,5X, '.../5X, ' THE SUBROUTINE CONSISTS OF A SEQUENCE OF BASIC, '
    * STATEMENTS ENDING WITH A 'RETURN, ' STATEMENT. ' THE SUBROUTINE, '
    * MAY ONLY BE ENTERED FROM A 'GOSUB, ' STATEMENT, ' AND WILL ONLY, '
    * RETURN TO ITS PROPER PLACE AFTER ENCOUNTERING A 'RETURN, ' STATEM
    * ENT, ' /5X, ' ALL THE VARIABLES OF THE MAIN PROGRAM ARE AVAILAB
    * LES, ' /5X, ' TO THE SUBROUTINE, ' AND VICE VERSA. ' THE VARIABLE
    * S, ' /5X, ' SUBROUTINE MUST BE CHOSEN CAREFULLY SO AS NOT TO ACCIDENTLY C
    * ONFLICT, ' /5X, ' OR ALTER VARIABLES IN THE MAIN PROGRAM. ' /5X, '
  305 READ(5,101,END=305)CARD
  105 WRITE(6,105)
    * B,C,5X, '.../5X, ' THE FOLLOWING PROGRAM TO COMPUTE THE GREATEST, '
    * COMMON DENOMINATOR (GCD) OF THREE NUMBERS BY EUCLID'S ALGORITHM, '
    * WILL DEMONSTRATE A USE OF SUBROUTINES. ' /5X, ' REM FIND GCD OF A, '
    * B,C,/5X, ' PRINT A,B,C,/5X, ' GCD, ' /3X, ' 20 READ A,B,C,/5X, '
    * IF C=9999 THEN 9999,/5X, ' LET X=A,/5X, ' LET Y=B,/5X, ' REM G=GCD
    * OF A,B,/5X, ' GOSUB 200,/5X, ' LET X=G,/5X, ' LET Y=C,/5X, ' REM G=GCD
    * LET X=Y,/14X, ' GOSUB 200,/5X, ' PRINT A,B,C,/5X, ' GO TO 20,/11X, ' 20
    * LET X=Y,/14X, ' LET R=X-Q*/14X, ' IF R=0 THEN 300,/14X, '
    * TURN,/5X, ' DATA 60,90,120,/5X, ' GO TO 200,/9X, ' 300 LET G=Y,/9X, ' RE
    * U,9999, ' DATA 38456,64872,98765,/5X, ' DATA 10X,B,10X, '
    * C,10X,GCD,/5X, ' 9999 END,/5X, ' OUTPUT PRODUCED : ' /5X, ' A, ' 10X, ' B, ' 10X, '
    * 72, ' 6X, ' 98765, ' 10X, ' 1, ' /5X, '
  306 READ(5,101,END=306)CARD
  106 WRITE(6,106)
    * B,C,5X, '.../5X, ' THE SUBROUTINE MAY BE NESTED TO LOGICALLY, '
    * CALL ONCE ANOTHER GOSUB, ' SIMILAR TO FOR/NEXT NESTED LOOPS, '
    * MUST ONCE MORE BE CAREFUL ABOUT THE STATUS OF VARIABLES BECAUSE, '
    * VARIABLES WILL BE PASSED FROM ONE SUBROUTINE TO ANOTHER. ' /5X, '
    * A SUBROUTINE THAT IS NESTED SO THAT IT CALLS ITSELF IS CALLED, '
    * RECURSION OF SUBROUTINES OR RECURSION. ' /5X, '
  307 READ(5,101,END=307)CARD
  107 WRITE(6,107)
    * B,C,5X, '.../5X, ' B. RECURSION TECHNIQUE IN WHICH SUBROUTINES CAN, '
    * RECURSION IS A PROGRAMMING TECHNIQUE IN WHICH SUBROUTINES CAN, '
    * CALL THEMSELVES. ' FOR EXAMPLE CONSIDER THIS PROGRAM WHICH FINDS, '
    * THE FACTORIAL OF A NUMBER USING ITERATIVE METHODS AND THEN, '
    * USING RECURSION : ' /5X, ' REM FACTORIAL : F(N)=1*2*3*...*(N-1)*N, '
    * 5X, ' PRINT, ' ITERATIVE SOLUTION, ' /6X, ' PRINT, ' N, ' , ' F(N), ' , ' /3X, ' , ' 10

```





















```

**FOLLOWED BY A DIGIT • EG A,A1,B6,Z0
**B) ALPHA VARIABLES : A SINGLE LETTER FOLLOWED BY A DOLLAR SIGN ($)
**/5X, • EG A$,V$
**C) SUBSCRIPTED VARIABLE : (SINGLE LETTER)
**((1) SINGLE SUBSCRIPT : <<LETTER>> ( <<EXPRESSION>> )
**((2) DOUBLE SUBSCRIPT : <<LETTER>> ( <<EXPRESSION,EXPRESSION>> ) /
**/ • EG A(5),Z(5,10),X(A*B,X**2),//(
  READ(5,101,END=303)CARD
303 WRITE(6,103)
103 FORMAT(0,103)
**A) EXPONENTIATION **
**B) DIVISION /
**C) MULTIPLICATION *
**D) ADDITION +
**E) SUBTRACTION -
**6. EXPRESSIONS : OR VARIABLE EG. 10,-.53,D,X(I,J)
**A) SINGLE NUMBER EG. SQR(10),TAN(.75)
**B) BUILT IN FUNCTION EG. SQR(10),TAN(.75)
**C) ARITHMETIC EXPRESSION : EXPRESSIONS SEPARATED BY OPERATORS
**AND GROUPED BY PARENTHESIS • EG. 5+6*.0 , A**2(3*X-4)
  READ(5,101,END=304)CARD
304 WRITE(6,104)
104 FORMAT(0,104)
**7. RELATIONS :
**A) GT A GREATER THAN B
**B) GE A GREATER THAN OR EQUAL TO B
**C) LT A LESS THAN B
**D) LE A LESS THAN OR EQUAL TO B
**E) NE A NOT EQUAL TO B
**F) EQ A EQUAL TO B
**8. BASIC STATEMENTS
**1. REM << ANY SET OF ALPHA-NUMERIC COMMENTS >>
**2. READ << VARIABLE,....,VARIABLE >>
**3. END
  READ(5,101,END=305)CARD
305 WRITE(6,105)
105 FORMAT(0,105)
**4. LET << VARIABLE >> = << EXPRESSION >>
**5. PRINT << STRING, • OR EXPRESSION,...., 'STRING, • OR EXPRESSION
**>> /5X, • OR SIMPLY PRINT
**6. DATA << STRING, • OR NUMBER,...., 'STRING, • OR NUMBER >>
**7. RESTORE << OR RESTORE$
**8. IF << EXPRESSION >> THEN << STATEMENT NUMBER >>
**/9. GO TO << EXPRESSION >>
**10. ON << EXPRESSION >> GO TO << STATEMENT NUMBER >>
**NUMBER >> //11. FOR << STATEMENT NUMBER,....,STATEMENT N
**TO << EXPRESSION >> STEP << EXPRESSION >> //30X
**//5X, • // NEXT << SIMPLE VARIABLE >>
**12. DIM << LETTER >> ( << INTEGER EXPRESSION, INTEGER EXPRESSION >>
**OR
**DIM << LETTER >> )

```



```

* )'/' 13. GOSUB << STATEMENT NUMBER >>
* '5X,'
* READ(5,101,'5X,' RETURN '//' )
* WRITE(6,106) CARD
* FORMAT(6,'C',) THAT CONCLUDES YOUR INSTRUCTION WITH CAI-BASIC '//'
* HOWEVER, DESIRE AT ANY TIME. IF YOU HAVE NOT RUN PROGRAMS UNDER '//'
* YOU DESIRE MODE (PUNCHING YOUR OWN CARDS AND HANDING THEM '//'
* THE OS/BATCH MODE (PUNCHING YOUR OWN CARDS AND HANDING THEM '//'
* ACROSS THE COUNTER TO BE RUN), YOU SHOULD GET THE BASIC MANUAL '//'
* TECHNICAL NOTE NR. 0211-12, IN ROOM I-147 TO FIND THE PROPER '//'
* JCB CONTROL CARDS REQUIRED. GOOD LUCK WITH THE COMPUTER, AND '//'
* REMEMBER IT ONLY DOES WHAT YOU TELL IT TO DO - GIGO (GARBAGE IN '//'
* CALL EXIT
* END

```

LES02030



## APPENDIX C

### 13.13.46 START EXECUTION BEGINS...

HI , WELCOME TO CAI-BASIC . THERE ARE ONLY A FEW  
SIMPLE RULES TO REMEMBER IN ORDER TO HAVE A SUCCESSFUL SESSION  
ON THE TERMINAL WITH CAI-BASIC :

1. WHEN ASKED FOR A RESPONSE , TYPE IN THE CORRECT REPLY AND  
HIT THE CARRIAGE RETURN KEY ON THE RIGHT SIDE OF THE KEYBOARD.

2. IF YOU MAKE A TYPING ERROR WHILE MAKING ANY RESPONSE  
OR INPUT , TYPE IN FOUR DOLLAR SIGNS (\$\$\$\$) AFTER THE ERROR OR  
ANYWHERE ON THAT INPUT LINE AND HIT CARRIAGE RETURN .THE  
ENTIRE LINE WILL THEN BE IGNORED AND YOU CAN TYPE IN THE CORRECT  
INPUT OR RESPONSE .

3. IF AT ANY POINT IN THE SESSION YOU WANT TO STOP THE SESSION  
TYPE IN THE WORD 'QUIT' AS SOON AS YOU ARE ASKED FOR THE NEXT  
RESPONSE ,HIT CARRIAGE RETURN,THEN HIT ATTN KEY AND TYPE LOGOUT .

4. DURING YOUR TERMINAL SESSION CAI-BASIC WILL HALT OCCASIONALLY  
TO LET YOU READ A SEQUENCE OF INFORMATION . WHEN YOU ARE  
READY TO CONTINUE , TYPE IN 'GO' , AND HIT CARRIAGE RETURN

5. DURING YOUR TERMINAL SESSION YOU MAY NOTICE THAT  
THE TYPING IS NOT ALWAYS PERFECT . SOME DAYS THE COMPUTER IS  
NOT UP TO PAR AND YOU WILL HAVE TO ADJUST TO THE MINOR IRRITANT

IF YOU ONLY WANT TO EXECUTE PROGRAMS AT THIS TIME THEN  
REPLY : YES ; OTHERWISE REPLY : NO .

no

IF THIS IS YOUR FIRST SESSION WITH CAIBASIC, THEN  
REPLY: YES ;OTHERWISE REPLY: NO .

yes

CAIBASIC IS A PROGRAM TO TEACH YOU THE  
FUNDAMENTALS OF A PROGRAMMING LANGUAGE.THE LANGUAGE TO BE LEARNED  
IS BASIC ; A SIMPLE LANGUAGE FOR THE USER WHO HAS LITTLE  
KNOWLEDGE OF COMPUTERS AND WHOSE PRIMARY INTEREST IS IN OBTAINING  
ESULTS.

THE SIMPLICITY OF THE BASIC LANGUAGE AND ITS RANGE  
OF CAPABILITIES SHOULD ALLOW YOU TO LEARN THE LANGUAGE AND



WRITE PROGRAMS IN A MINIMAL AMOUNT OF TIME.

THE REFERENCE TEXTS RECOMMENDED FOR CAIBASIC ARE :

1. BASIC LANGUAGE MANUAL , TN # 0211-12 APRIL 1971  
( FREE UPON REQUEST IN I-147 )

2. INTRODUCTION TO COMPUTING THROUGH THE BASIC LANGUAGE , R.L. NOLAN  
(BOOKSTORE / MAIN LIBRARY)

3. BASIC PROGRAMMING , V.C. HARE  
(COMPUTER CENTER LIBRARY )

\*\*\* AFTER YOU HAVE FINISHED READING AN INPUT , TYPE IN GO AND HIT THE RETURN  
AND THE PROGRAM WILL CONTINUE \*\*\*

go

DURING YOUR TERMINAL SESSION YOU WILL BE LEARNING  
THE STRUCTURE OF THE LANGUAGE BASIC. THE INSTRUCTION SET  
CONTAINS 7 LESSONS AND YOU MAY PROCEED THROUGH THE LESSONS AT  
YOUR OWN SPEED.

THE LESSONS ARE AS FOLLOWS :

LESSON 1 PROGRAM FORMAT AND BASIC DEFINITIONS  
LESSON 2 REMARKS, INPUT/OUTPUT AND DATA  
LESSON 3 ASSIGNMENT STATEMENTS AND BUILT IN FUNCTIONS  
LESSON 4 BRANCHING  
LESSON 5 LOOPING AND SUBSCRIPTED VARIABLES  
LESSON 6 SUBROUTINES AND RECURSION  
LESSON 7 SUMMARY OF BASIC STATEMENTS

\*\* WHEN YOU ARE READY TO CONTINUE , TYPE IN GO AND HIT  
THE CARRIAGE RETURN \*\*\*

go

IN EACH LESSON YOU WILL BE GIVEN INSTRUCTION  
SEQUENCES AND THEN YOU WILL BE ASKED QUESTIONS TO SEE IF YOU  
UNDERSTOOD THE INSTRUCTIONS. THE QUESTIONS WILL BE OF VARIOUS  
TYPES: MULTIPLE CHOICE, TRUE/FALSE, ACTUAL PROGRAM STATEMENTS, ETC.  
YOU WILL BE PROMPTED FOR YOUR ANSWER, AND WHEN READY TYPE IN  
OUR RESPONSE AND HIT THE RETURN KEY.

IF YOU KEEP THE TELETYPE OUTPUT FROM YOUR TERMINAL SESSION  
YOU WILL HAVE A READY REFERENCE FOR FUTURE USE





go

\*\*\* LESSON 1. \*\*\*

THIS INSTRUCTION SET WILL INTRODUCE YOU TO THE STRUCTURE OF BASIC LANGUAGE STATEMENTS , THE RULES FOR VARIABLES AND NUMBERS , AND THE SYMBOLS FOR ARITHMETIC OPERATIONS.

\*\*\*DONT FORGET TO TYPE GO AND HIT RETURN WHEN READY TO CONTINUE . \*\*\*

go

A. PROGRAM STRUCTURE

THE BASIC LANGUAGE , AND ALL OTHER LANGUAGES , HAS A SPECIFIED STRUCTURE . EACH BASIC STATEMENT HAS A REQUIRED FORM WITH POSSIBLY ONE OR MORE VARIATIONS. THE FOLLOWING EXAMPLE WILL ILLUSTRATE SOME SIMPLE BASIC STATEMENTS IN THE PROPER PROGRAM STRUCTURE

```
REM PROGRAM TO COMPUTE GAS MILAGE
REM M=MILES TRAVELED , G=GAS USED
READ M,G
LET T=M / G
PRINT 'MILES TRAVELED','GAS USED',' MILES/GAL'
PRINT M,G,T
DATA 500,25
END
```

OUTPUT PRODUCED IS :

| MILES TRAVELED | GAS USED | MILES/GAL |
|----------------|----------|-----------|
| 500            | 25       | 20        |

go

AS YOU CAN SEE FROM THE ABOVE SAMPLE, THE PROGRAM STRUCTURE CONSISTS OF BASIC LANGUAGE STATEMENTS FOLLOWED BY AN END STATEMENT . THE WORDS : REM , READ, LET , PRINT , DATA AND END ARE KEY WORDS THAT MAKE UP A BASIC STATEMENT .

go

MOST OF THE KEY WORDS USED IN THE BASIC STATEMENTS ARE SELF-EXPLANATORY:

REM ALLOWS REMARKS/COMMENTS  
READ M,G ASSIGNS NUMBERS IN THE DATA STATEMENT TO THE VARIABLES M AND G  
LET ASSIGNS THE RESULT OF M DIVIDED BY G INTO VARIABLE T



PRINT'String' CAUSES THE STRING IN SINGLE QUOTES TO BE  
PRINTED LITERALLY  
PRINT M,G,T PRINTS THE CURRENT VALUE OF THE VARIABLES M,G,T  
END TELLS THE COMPUTER THAT THE INPUT PROGRAM IS TO BE  
EXECUTED

go

IF AT THIS POINT YOU WOULD LIKE TO RUN THE  
SAMPLE PROGRAM TO GAIN SOME CONFIDENCE IN THE COMPUTER AND ITS  
ABILITY TO PROVIDE SPEEDY RESULTS, THEN REPLY : YES ; OTHER-  
WISE REPLY : NO AND THE INSTRUCTION WILL CONTINUE .

no

## B. PROGRAM FORMAT

A BASIC PROGRAM CONSISTS OF A SEQUENCE OF BASIC STATEMENTS , ONE  
STATEMENT PER INPUT LINE , FOLLOWED BY AN END STATEMENT .  
BECAUSE NO BASIC STATEMENT MAY BE LONGER THAN ONE INPUT LINE (80 SPACES) ,  
THERE IS NO PROVISION FOR CONTINUING STATEMENTS FROM ONE LINE  
TO THE NEXT . HOWEVER ; YOU MAY SPACE THE INPUT LINE  
AS DESIRED FOR READABILITY SINCE THE COMPUTER IGNORES BLANKS IN BASIC .

go

### 1. STATEMENT NUMBERS

EACH BASIC STATEMENT MAY HAVE AN OPTIONAL  
STATEMENT NUMBER PRECEDING IT FOR IDENTIFICATION PURPOSES .  
THIS STATEMENT NUMBER MUST BE AN INTEGER BETWEEN 1 -> 9999  
FOR EXAMPLE : 12 READ M,G

### 2. KEY WORDS

THE KEY WORDS THAT MAKE UP A BASIC STATEMENT (REM ,READ ,LET ,ETC,)  
ARE SPECIAL TERMINAL SYMBOLS THAT ARE RECOGNIZED BY THE COMPUTER  
AND FOR THIS REASON THEY MUST BE SPELLED CORRECTLY AND ONLY  
USED IN BASIC STATEMENTS .

THE END STATEMENT INDICATES THAT THE INPUT PROGRAM IS  
COMPLETED AND THAT PROGRAM EXECUTION IS TO BEGIN . THE 'END'  
STATEMENT IS ALWAYS THE LAST STATEMENT IN A PROGRAM .

go

YOU WILL NOW BE ASKED A FEW SIMPLE QUESTIONS ABOUT  
WHAT YOU HAVE JUST LEARNED .

1.) IS THIS A LEGAL BASIC PROGRAM(REPLY : YES OR NO)??



```
REM ONE LINE DO NOTHING PROGRAM
END
```

yes

YES , THE SIMPLEST BASIC PROGRAM CONSISTS OF JUST AN 'END' STATEMENT.

2.) WHICH OF THE FOLLOWING BASIC STATEMENTS IS  
IN THE PROPER FORMAT :

- A.) 15 LET T=M/G
- B.) 15LETT=M/G
- C.) 15 LET T = M / G

REPLY A , B , C OR ALL .

all

ALL OF THE ABOVE BASIC STATEMENTS ARE CORRECT BECAUSE  
SPACES ARE DISREGARDED . NOTE THAT STATEMENT B.) , ALTHOUGH  
CORRECT IS CONFUSING TO READ . BLANKS AND INDENTATIONS MAKE A  
PROGRAM EASY TO READ .

#### C. ALPHA-NUMERIC CHARACTERS

ALPHA-NUMERIC CHARACTERS ARE THE LEGAL CHARACTERS , DIGITS , AND  
SPECIAL CHARACTERS THAT CAN BE USED IN BASIC .

##### 1. CHARACTERS

CHARACTERS CONSIST OF THE LETTERS IN THE ALPHABET A-->Z

##### 2. DIGITS

DIGITS ARE THE SINGLE NUMBERS 0-->9

##### 3. SPECIAL CHARACTERS

THE SPECIAL CHARACTERS ARE \* \* / + - ( ) = ' , \$

##### 4. STRING

A STRING IS ANY LIST OF ALPHA-NUMERIC CHARACTERS  
ENCLOSED IN 'SINGLE' QUOTES . FOR EXAMPLE :

'THIS IS A STRING'

go





## D. VARIABLES

IN BASIC THERE ARE THREE TYPES OF VARIABLES : SIMPLE , ALPHA AND SUBSCRIPTED . SUBSCRIPTED VARIABLES WILL BE COVERED IN LESSON 5 .

### 1. SIMPLE VARIABLES

SIMPLE VARIABLES ARE IDENTIFIED BY A SINGLE LETTER OR A SINGLE LETTER FOLLOWED BY A DIGIT BETWEEN 0->9 .

FOR EXAMPLE : A , A3 , Z0 , AND X ARE LEGAL VARIABLES ; BUT A26 , 9Z , ABC , AND XIY ARE ILLEGAL VARIABLES . THEREFORE ; YOU HAVE 286 SIMPLE VARIABLES FOR USE IN YOUR PROGRAMS.

go

### 2. ALPHA VARIABLES

ALPHA VARIABLES ARE USED FOR ALPHA-NUMERIC MANIPULATIONS IN WHICH A GROUP OF ALPHA-NUMERIC CHARACTERS , CALLED A STRING , ARE REPRESENTED BY AN ALPHA VARIABLE . THE ALPHA VARIABLE CONSISTS OF A SINGLE LETTER FOLLOWED BY A DOLLAR SIGN , \$ . THE MAXIMUM LENGTH OF THE ALPHA-NUMERIC STRING ASSIGNED TO THE ALPHA VARIABLE IS 16 CHARACTERS.

FOR EXAMPLE : A\$ , X\$ , Z\$ ARE LEGAL ALPHA VARIABLES. A SIMPLE USE OF ALPHA VARIABLES FOLLOWS :

```
READ A$,B$
PRINT A$,B$
DATA 'MONDAY','21 JUNE'
END
```

THIS PROGRAM WILL PRODUCE THE OUTPUT :

MONDAY 21 JUNE

go

## E. NUMBERS

NUMBERS MAY BE EXPRESSED AS INTEGERS OR AS REALS ,AND A NUMBER WHETHER INTEGER OR REAL IS LIMITED TO 9 DIGITS , NOT INCLUDING DECIMAL POINT. A NUMBER IS ASSUMED POSITIVE UNLESS IT IS PRECEDED BY A - SIGN .

### 1. INTEGERS

INTEGERS ARE NUMBERS WITH NO FRACTIONAL PART ,IE. 3 , 15



## 2. REALS

REAL (FIXED-POINT) NUMBERS HAVE A DECIMAL POINT AND A FRACTIONAL PART , I.E. 3.0 , 15.31, 729.1 , 0.0

go

### F. EXPRESSIONS

AN EXPRESSION MAY BE A SINGLE CONSTANT OR VARIABLE , OR AN ARITHMETIC EXPRESSION . ARITHMETIC EXPRESSIONS ARE FORMED BY USING OPERATORS AND PARENTHESIS .

#### 1. OPERATORS

NUMBERS AND SIMPLE VARIABLES MAY BE COMBINED INTO ARITHMETIC EXPRESSIONS BY USING ONE OR MORE OF THE ARITHMETIC OPERATORS :

- \*\* EXPONENTIATION
- \* MULTIPLICATION
- / DIVISION
- + ADDITION
- SUBTRACTION

IN WRITING AN EXPRESSION , EACH ARITHMETIC OPERATION MUST BE SHOWN. NO TWO OPERATIONS MAY BE ADJACENT , NOR MAY TWO NUMBERS OR VARIABLES BE ADJACENT IN AN EXPRESSION. FOR EXAMPLE:  $2+2$  ,  $A/B$  ,  $B+C$  ,  $X*Z$  , AND  $Y-3.0$  ARE LEGAL EXPRESSIONS.

go

#### 2. PARENTHESIS

PARENTHESIS MAY BE USED TO GROUP EXPRESSIONS AND TO CONTROL THE ORDER IN WHICH AN ARITHMETIC EXPRESSION IS EVALUATED . THE NORMAL HIERARCHY OF OPERATORS IS :

1. \*\*
2. \* AND /
3. + AND -

IF PARENTHESIS ARE USED , THE EXPRESSION WITHIN A PARENTHESIS PAIR IS EVALUATED FIRST . PARENTHESIS IN EXPRESSIONS MUST ALWAYS OCCUR IN PAIRS , I.E.  $5 * (4 + 3)$  ,  $(5*(4+3))$  ARE EQUIVALENT EXPRESSIONS HAVING THE VALUE 35 .

OPERATIONS ARE EVALUATED ACCORDING TO THE OPERATORS HIERARCHY IF TWO OPERATORS OF THE SAME HIERARCHY OCCUR IN AN EXPRESSION EVALUATION IS FROM LEFT TO RIGHT . FOR EXAMPLE:  $2*4 + 6/3$

IN THIS EXPRESSION  $2*4$  WILL BE EVALUATED FIRST , THEN  $6/3$  , AND THEN THE TWO RESULTS WILL BE ADDED .



go

YOU WILL NOW BE ASKED A FEW SIMPLE QUESTIONS ABOUT  
WHAT YOU HAVE JUST LEARNED .

1.)  $4 + 6 / 2$  HAS THE VALUE (REPLY WITH VALUE)

5

YOUR ANSWER IS INCORRECT . THE EXPRESSION IS EVALUATED AS FOLLOWS :

$4+6/2 \rightarrow 4+3 \rightarrow 7$

2.)  $(4+6)/2$  HAS THE VALUE (REPLY WITH VALUE)

5

3.)  $((4+6)/2)**2$  HAS THE VALUE (REPLY : VALUE)

25

#### F. SUMMARY

THIS CONCLUDES THE INSTRUCTION SET FOR LESSON 1.  
YOU WILL NOW BE GIVEN A SAMPLE BASIC PROGRAM AND ASKED TO FIND  
THE MISTAKES IN IT , CONCERNING WHAT YOU HAVE LEARNED IN THIS  
LESSON :

```
1 REM REVIEW PROGRAM
2 READ A1,B2,Z1,$Y
3 LET A1=B2(A1+Z1)
4 LET B2=((A1*3)+Z1**2)
5 LET Z1=3.141592763
6 PRINT 1A,B2,Z1
7 DATA 5.0,3,3.1416,'FINI'
```

AFTER LOOKING AT THE SAMPLE PROBLEM , YOU WILL BE ASKED QUESTIONS  
ABOUT EACH STATEMENT ,

go

1.) LINE 2 CONTAINS A READ STATEMENT FOLLOWED  
BY A LIST OF VARIABLES . ANY ERRORS (REPLY: YES OR NO)?

yes



INPUT CORRECTION TO THE INCORRECT VARIABLE ONLY .

y\$

2.) LINE 3 CONTAINS A LET STATEMENT FOLLOWED  
BY A1 = EXPRESSION . ANY ERRORS (REPLY : YES OR NO) ?

yes

INPUT CORRECTION TO ILLEGAL EXPRESSION ; EVERYTHING  
AFTER = SIGN .

b2\*(a1+z1)

3.) LINE 4 IS SIMILAR TO LINE 3 . ANY ERRORS  
REPLY YES OR NO) ?

no

4.) LINE 5 CONTAINS A LET STATEMENT FOLLOWED  
BY A NUMBER . ANY ERRORS (REPLY : YES OR NO) ?

no

THE NUMBER CONTAINS 10 DIGITS AND THE MAXIMUM ALLOWED  
IS 9 DIGITS.

5.) LINE 6 CONTAINS A PRINT STATEMENT FOLLOWED  
BY A LIST OF VARIABLES . ANY ERRORS (REPLY : YES OR NO)?

yes

INPUT CORRECTION TO THE INCORRECT VARIABLE ONLY .

a1

THERE ARE NO ERRORS IN LINE 7 . IS THE PROGRAM  
READY TO EXECUTE (ASSUMING ABOVE ERRORS CORRECTED)  
(REPLY : YES OR NO)

yes

THE PROGRAM FORMAT REQUIRES THAT AN END STATEMENT





BE THE LAST STATEMENT OF THE PROGRAM . THUS THIS SAMPLE PROGRAM  
WOULD NOT EXECUTE .

THIS CONCLUDES THE REVIEW QUESTIONS  
FROM LESSON 1 .

DO YOU WANT TO TERMINATE YOUR INSTRUCTION SESSION ?  
(REPLY : YES OR NO )

no

IF YOU WANT TO EXECUTE PROGRAMS AT THIS TIME  
THEN REPLY : YES ; OTHERWISE REPLY : NO AND THE INSTRUCTION  
WILL CONTINUE .

no



EXECUTION BEGINS...

\*\*\* LESSON 2. \*\*\*

THIS INSTRUCTION SEQUENCE WILL INTRODUCE YOU TO THE BASIC LANGUAGE STATEMENTS REM , PRINT , READ , AND DATA . USING THESE STATEMENTS YOU WILL BE ABLE TO CONSTRUCT AND EXECUTE ELEMENTARY PROGRAMS .  
THE FORM FOR EACH BASIC STATEMENT WILL INCLUDE :

'KEY WORD' << ELEMENTS OR LIST OF ELEMENTS SEPARATED BY COMMAS >>

THE CARET SYMBOLS '<< >>' DELINIATE THE LEGAL ITEMS THAT MAY FOLLOW THE KEYWORD AND MAKE UP THE BASIC STATEMENT .

go

A. REM

THE BASIC STATEMENT WHICH ALLOWS YOU TO INSERT REMARKS INTO YOUR PROGRAM IS IDENTIFIED BY THE KEY WORD 'REM'. REM IS A NON-EXECUTING STATEMENT WHICH MAY BE USED OPTIONALLY AT ANY PLACE IN YOUR PROGRAM TO INTRODUCE A PROGRAM NAME , TO EXPLAIN VARIABLES, TO DOCUMENT YOUR PROGRAM , ETC.

go

THE FORM FOR THE REM STATEMENT IS :

REM << ANY SEQUENCE OF ALPHA-NUMERIC CHARACTERS >>

THE REM STATEMENT IS IGNORED BY THE CAIBASIC COMPILER , AND IS ONLY FOR YOUR INFORMATION . FOR EXAMPLE :

REM PROGRAM TO COMPUTE INCOME TAX

go

B. PRINT

THE PRINT STATEMENT IS THE METHOD OF WRITING OUT THE RESULTS OF THE BASIC PROGRAM , TO DISPLAY VALUES OF VARIABLES , TO LABEL STATEMENT IS :

PRINT << EXPRESSION OR 'STRING',...,EXPRESSION OR 'STRING' >>

MULTIPLE ELEMENTS IN THE PRINT LIST ARE SEPARARED BY COMMAS.

go

1. PRINT << EXPRESSION >> WILL WRITE OUT THE CURRENT VALUE OF THE EXPRESSION , WHERE AN EXPRESSION AS DEFINED IN LESSON 1 WAS A NUMBER , A VARIABLE , OR AN ARITHMETIC



EXPRESSION THAT IS TO BE EVALUATED .  
FOR EXAMPLE :

```
PRINT 1,A,B1,5**2
```

ASSUMING THAT A=10.0 , B=13.3 WOULD PRINT :

|   |      |      |    |
|---|------|------|----|
| 1 | 10.0 | 13.3 | 25 |
|---|------|------|----|

go

2. PRINT <<'STRING' >> WILL PRINT OUT ALL THE ALPHA-NUMERIC CHARACTERS OF THE STRING WITHIN 'SINGLE' QUOTES. THIS FORM IS USED FOR LABELING THE COMPUTER OUTPUT.

FOR EXAMPLE : PRINT 'THE ANSWER IS :'

PRODUCES THE RESULT :

THE ANSWER IS :

3. PRINT BY ITSELF IS USED TO SKIP A LINE ON THE COMPUTER OUTPUT .

go

THE COMPUTER OUTPUT SHEET IS DIVIDED INTO 8 ZONES EACH 15 COLUMNS WIDE . PRINT ZONES CAN BE SKIPPED BY PUTTING A BLANK IN THE PRINT LIST . FOR EXAMPLE :

```
PRINT A, B, ,X
```

OUTPUTS THE VALUE OF A IN THE FIRST ZONE , B IN THE SECOND , SKIPS THE THIRD ZONE , AND PUTS X IN THE FOURTH ZONE .

ALPHA VARIABLES AND 'STRINGS' MAY EXTEND OVER SEVERAL ZONES , BUT NUMERIC RESULTS ARE LEFT ADJUSTED IN THE SPECIFIED ZONE . IF MORE THAN EIGHT ITEMS OCCUR IN THE PRINT LIST , THE ITEMS WILL OVERFLOW AND BE PRINTED ON THE NEXT LINE .

go

USING THE PRINT AND END STATEMENT YOU NOW HAVE THE FACILITY TO EXECUTE YOUR FIRST PROGRAMS . FOR EXAMPLE :

```
REM PROGRAM TO COMPUTE THE SQUARE OF A NUMBER  
PRINT '5 SQUARED =',5**2  
END
```

PRODUCES THE RESULT :

5 SQUARED = 25

YOU WILL NOW BE GIVEN TWO PROBLEMS TO SOLVE . YOU WILL ENTER





THE EXECUTION PHASE OF CAIBASIC WHERE YOU CAN RUN YOUR PROBLEMS  
AND THEN YOU WILL RETURN TO FINISH THE LESSON.

1) FIND THE SQUARE ROOT OF 5 SQUARED MINUS 4 TIMES 2 TIMES 2 .

2) FIND THE VALUE OF  $3.1416(B \text{ CUBED})H/12$   
WHERE  $B=2.50$  ,  $H=3.03$  .

IF YOU WISH TO SKIP THESE PROBLEMS REPLY : YES  
AND THE LESSON WILL CONTINUE.

yes

#### C. READ

THE READ STATEMENT IS THE METHOD WHICH PROVIDES INPUT TO THE  
PROGRAM . THE FORM OF THE READ STATEMENT IS :

READ << VARIABLE,.....,VARIABLE >>

FOR EVERY VARIABLE IN THE READ LIST THERE MUST BE A CORRESPONDING  
ELEMENT IN A DATA STATEMENT . THE READ AND DATA STATEMENTS ARE  
USED TOGETHER TO ASSIGN INPUT VALUES TO PROGRAM VARIABLES.

WHEN THE READ STATEMENT IS EXECUTED , EACH VARIABLE IS ASSIGNED  
SUCCESSIVE NUMBERS FROM A STACK OF NUMERIC DATA OR SUCCESSIVE  
'STRINGS' FROM A STACK OF ALPHA-NUMERIC DATA. AS EACH VARIABLE  
IS READ , IT TAKES THE TOP ELEMENT OF THE APPROPRIATE DATA  
STACK .

go

#### D. DATA

THE DATA STATEMENT IS A LIST OF INPUT NUMBERS OR 'STRINGS' THAT  
WILL BE ASSIGNED TO VARIABLES IN A READ STATEMENT . THE FORM  
OF THE DATA STATEMENT IS :

DATA << NUMBER OR 'STRING',....., NUMBER OR 'STRING' >>

THE 'STRING' OF ALPHA-NUMERIC CHARACTERS MUST BE ENCLOSED IN  
SINGLE QUOTES .

DATA STATEMENTS MAY BE PLACED ANYWHERE IN A PROGRAM , BUT THERE  
IS AN UPPER LIMIT OF 500 NUMERIC AND 500 ALPHA-NUMERIC DATA  
ELEMENTS FOR EACH PROGRAM .

go



WHEN THE FIRST DATA STATEMENT IS INTERPRETED BY THE CAIBASIC COMPILER A FIRST IN , FIRST OUT STACK IS FORMED FOR NUMERIC AND ALPHA-NUMERIC DATA . AS EACH NUMBER OR STRING IN A DATA LIST IS INTERPRETED , IT IS PLACED ON THE BOTTOM OF ITS RESPECTIVE DATA STACK . AS OTHER DATA STATEMENTS ARE LOCATED IN THE PROGRAM ITS ELEMENTS ARE PLACED ON THE BOTTOM OF THE PROPER STACK .

#### 1.) EXAMPLE :

```
DATA 10.0,13.33,'J.E.SMITH',18
DATA 'Z.X. DOE',19
```

PRODUCES THE FOLLOWING DATA STACKS :

| NUMERIC | ALPHA-NUMERIC |
|---------|---------------|
| 10.0    | J.E.SMITH     |
| 13.33   | Z.X. DOE      |
| 18      |               |
| 19      |               |

go

DURING EXECUTION OF READ STATEMENTS , AS THE VARIABLES IN THE READ LIST ARE ASSIGNED VALUES FROM THE TOP OF THE APPROPRIATE DATA STACK , THE DATA STACK IS DECREMENTED AND THE NEXT ELEMENT POPS UP .

#### 2.) EXAMPLE :

```
READ A,B1,Z$
```

ASSUMING THAT THE DATA FROM EXAMPLE 1.) IS AVAILABLE , THE VARIABLES IN THE READ LIST ARE ASSIGNED VALUES AS FOLLOWS :

```
A <-- 10.0
B1 <-- 13.33
Z$ <-- J.E.SMITH
```

THE RESULTING DATA STACKS ARE AS FOLLOWS :

| NUMERIC | ALPHA-NUMERIC |
|---------|---------------|
| 18      | Z.X. DOE      |
| 19      |               |

go

#### E. RESTORE , RESTORE\$

THE RESTORE , RESTORE\$ STATEMENTS ARE USED TO RETURN THE NUMERIC AND ALPHA-NUMERIC DATA STACKS TO THEIR ORIGINAL CONDITION SO



7.1  
THAT THE DATA MAY BE USED AGAIN . THE FORM OF THE RESTORE STATEMENT  
IS :

RESTORE ( RESTORES NUMERIC DATA )

RESTORE\$ ( RESTORES ALPHA-NUMERIC DATA )

YOU WILL NOW BE ASKED SOME QUESTIONS ABOUT THE READ , DATA AND  
RESTORE STATEMENTS :

CONSIDER THE FOLLOWING STATEMENTS AS PART OF A BASIC PROGRAM:

```
DATA 2,5,7,16,'ANS='  
DATA 'CORRECT',25,'WRONG',0.0  
READ A,B3,I$,C1,D  
READ J$,E4,K$,X
```

go

1.) WHAT IS THE VALUE OF C1 ?? REPLY WITH VALUE

7

NOW CONSIDER THAT THE FOLLOWING STATEMENTS  
WERE ADDED TO THE ABOVE PROGRAM :  
RESTORE  
READ Z4,A

2.) WHAT IS THE VALUE OF Z4 ?? REPLY WITH VALUE .

16

THE CORRECT ANSWER IS Z4=2 . THE RESTORE COMMAND  
RETURNS THE NUMERIC DATA STACK TO ITS ORIGINAL CONDITION, AND  
THE READ STATEMENT ASSIGNS VALUES FROM THE TOP OF THE DATA  
STACK TO THE VARIABLES IN THE READ LIST AS FOLLOWS :

```
Z4 <-- 2  
A <-- 5
```

#### F. SUMMARY

NOW THAT YOU HAVE SEEN HOW READ AND DATA STATEMENTS WORK TOGETHER  
TO INPUT VALUES INTO YOUR PROGRAM , AND HOW THE PRINT STATEMENT  
IS USED TO OUTPUT AND LABEL RESULTS YOU HAVE THE FACILITY  
TO WRITE SIMPLE PROGRAMS USING INPUT DATA .

FOR EXAMPLE

(( 8\*\*2)\*\*.5) COULD BE WRITTEN IN SYMBOLIC FORM AND THE DATA



COULD BE READ IN AS FOLLOWS:

```
READ A,B
PRINT'ANS=',(( A**B)**.5)
DATA 8,2
END
```

IN THE NEXT LESSON YOU WILL BE SHOWN HOW TO USE ASSIGNMENT STATEMENTS  
THIS WILL GIVE YOU GREATER FLEXIBILITY IN WRITING EXPRESSIONS  
AND WILL ALLOW YOU TO DO ASSIGNMENTS SUCH AS :

$A=B**2-4*A*C$  , AND  $Z=(A**2+3)/A$  , THEN BY SAYING PRINT A,Z  
THE RESULTS OF BOTH EXPRESSIONS WOULD BE DISPLAYED.

go

YOU WILL NOW BE GIVEN SOME REPRESENTATIVE PROBLEMS  
TO GIVE YOU A CHANCE TO EXERCISE YOUR NEW PROGRAMMING TOOLS

1.) WRITE A PROGRAM TO SOLVE THE EQUATION  $X**2+10Y-24$   
WHERE THE INPUT DATA IS  $X=10$  ,  $Y=3$  .

2.) WRITE A PROGRAM TO SOLVE THE QUADRATIC EQUATION

$(-B+(B**2-4AC)**.5)/2A$   
INPUT DATA IS  $A=2$  ,  $B=5$  ,  $C=2$  .

IF YOU WISH TO WRITE AND EXECUTE THESE PROGRAMS NOW , REPLY : YES  
AND YOU WILL ENTER THE CAIBASIC COMPILER ; OTHERWISE REPLY : NO  
AND YOU WILL GO ON TO THE NEXT LESSON .

no

IF YOU DECIDE TO RUN THESE PROBLEMS LATER THE ANSWERS  
WILL NOW BE GIVEN SO YOU MAY CHECK YOUR RESULTS :

1.) 106  
2.) -.500

DO YOU WANT TO TERMINATE YOUR INSTRUCTION SESSION ?  
(REPLY : YES OR NO )

no

IF YOU WANT TO EXECUTE PROGRAMS AT THIS TIME  
THEN REPLY : YES ; OTHERWISE REPLY : NO AND THE INSTRUCTION  
WILL CONTINUE .

no





EXECUTION BEGINS...

\*\*\* LESSON 3. \*\*\*

THIS INSTRUCTION SET WILL INTRODUCE YOU TO ASSIGNMENT STATEMENTS AND BUILT-IN FUNCTIONS. THE LET STATEMENT AND THE 10 BUILT IN FUNCTION WILL ENABLE YOU TO EVALUATE AND ASSIGN VARIABLES TO COMPLEX ARITHMETIC EXPRESSIONS.

go

A. LET

THE LET STATEMENT IS AN ASSIGNMENT OR SUBSTITUTION COMMAND. IT CAUSES THE EVALUATION OF AN EXPRESSION TO BE SUBSTITUTED FOR THE CURRENT VALUE OF A VARIABLE. THE FORM OF THE LET STATEMENT IS:

LET << VARIABLE >> = << EXPRESSION >> OR

LET << VARIABLE >> = << VARIABLE >> = .... = << EXPRESSION >>

go

THERE MAY BE ANY NUMBER OF VARIABLE = VARIABLE IN THE FORM OF THE LET STATEMENT. AN EXPRESSION IS A NUMBER, VARIABLE, OR AN ARITHMETIC EXPRESSION.

WHEN THE LET STATEMENT IS EXECUTED THE EXPRESSION ON THE RIGHT SIDE OF THE EQUAL SIGN IS EVALUATED AND THE RESULTING VALUE IS ASSIGNED TO THE VARIABLE OR VARIABLES ON THE LEFT SIDE OF THE EQUAL SIGN. THE PREVIOUS VALUE ASSIGNED TO THE VARIABLE OR VARIABLES WILL BE LOST. FOR EXAMPLE:

LET A=12.3  
LET B=14.4  
LET A=B=25

THE VALUE OF THE VARIABLES A AND B IS NOW A=B= 25 .

go

THE ONLY RESTRICTION ON THE USE OF VARIABLES IS THAT SIMPLE AND SUBSCRIPTED VARIABLES CAN ONLY BE ASSIGNED NUMERIC VALUES, AND ALPHA VARIABLES CAN ONLY BE ASSIGNED ALPHA-NUMERIC STRINGS. FOR EXAMPLE:

LET A=D4=X(5)=(3\*\*2 - 6)/4  
LET A\$=X\$='HELP'



```

      LET Y=(B - 4*A*C)**.5
      LET O$='ANSWER ='
THE FOLLOWING ASSIGNMENT IS ILLEGAL :

```

```

      LET A=D$(3+4)

```

80

YOU WILL NOW BE ASKED QUESTIONS CONCERNING  
WHAT YOU HAVE JUST LEARNED :

1.) REPLY WITH VALUE OF X IN BELOW PROGRAM .

```

      LET A=4
      LET B=6
      LET C=3
      LET X=A**2 - 4*B + 3*C
      PRINT 'ANSWER =' , X
      END

```

1

2.) REPLY WITH VALUE OF Z IN BELOW PROGRAM .

```

      DATA 1,2,3,4,6
      DATA 'RIGHT ON',7,8
      READ A,B,X
      READ C$,Y,C,D
      LET Z=X+Y
      END

```

6

YOUR RESPONSE WAS INCORRECT . THE VARIABLES ARE  
ASSIGNED VALUES AS FOLLOWS :

|         |                 |
|---------|-----------------|
| NUMERIC | ALPHA           |
| A<-- 1  | Y\$<-- RIGHT ON |
| B<-- 2  |                 |
| X<-- 3  |                 |
| Y<-- 4  |                 |
| C<-- 6  |                 |
| D<-- 7  |                 |

```

      Z <-- X+Y      ;      Z <-- 7

```

3.) LET R= A+B/C-D  
WHICH FORMULA DOES THIS STATEMENT REPRESENT ?



R  
EPLY WITH CORRECT LETTER  
A.)  $R=(A+B)/(C-D)$   
B.)  $R= A+(B/C)-D$

b

B. BUILT-IN FUNCTIONS  
BUILT-IN FUNCTIONS ARE COMMONLY USED PROGRAMS ALREADY WRITTEN  
AND STORED IN THE CAIBASIC COMPILER FOR YOUR USE . THERE ARE  
FUNCTIONS TO FIND SQUARE ROOTS , LOGARITHMS , ABSOLUTE VALUES ,  
AND TRIGONOMETRIC VALUES . THE FORM FOR THE BUILT-IN FUNCTIONS IS :

FUNCTION NAME << (EXPRESSION) >>  
WHERE THE EXPRESSION IS ENCLOSED IN PARENTHESIS .

go

THE BUILT-IN FUNCTIONS AND DEFINITIONS ARE :

|        |                                                     |
|--------|-----------------------------------------------------|
| SQR(X) | --- SQUARE ROOT OF ARGUMENT (MUST BE POSITIVE)      |
| ABS(X) | --- ABSOLUTE VALUE OF ARGUMENT                      |
| LOG(X) | --- NATURAL LOGARITHM OF ARGUMENT                   |
| EXP(X) | --- EXPONENTIAL FUNCTION , VALUE OF $2.718218 ** X$ |
| INT(X) | --- INTEGER PART OF ARGUMENT IS RETURNED            |
| SIN(X) | --- SINE OF THE ARGUMENT                            |
| COS(X) | --- COSINE OF ARGUMENT                              |
| TAN(X) | --- TANGENT OF ARGUMENT                             |
| ATN(X) | --- ARCTANGENT IN RADIANS OF ARGUMENT               |

IN ALL THE ABOVE BUILT-IN FUNCTIONS THE ARGUMENT IS ANY LEGAL EXPRESSION ;  
AND AS NOTED THE SQR FUNCTION REQUIRES A POSITIVE ARGUMENT ,  
THE TRIGONOMETRIC FUNCTIONS REQUIRE AN ARGUMENT VALUE IN RADIANS .

go

THE BUILT-IN FUNCTIONS ARE USED BY SIMPLY  
CALLING THEM WITH THE APPROPRIATE FUNCTION NAME AND ARGUMENT .  
THESE BUILT-IN FUNCTIONS ARE CONSIDERED TO BE EXPRESSIONS ,  
AND THEY MAY BE USED ANY PLACE WHERE AN EXPRESSION IS LEGAL ,  
FOR EXAMPLE :

LET Z=SQR(ABS(-5))      IS A CORRECT USE OF BUILT IN FUNCTIONS.

go



THE FOLLOWING PROGRAM IS A EXAMPLE OF HOW TO  
USE BUILT-IN FUNCTIONS :

```
REM PROGRAM TO COMPUTE SQUARE ROOT AND LOGARITHMS
READ A
LET Y=SQR(A)
LET Z=LOG(A)
PRINT 'SQUARE ROOT=',Y,'LOG=',Z
DATA 4,6,8
END
```

PRODUCES RESULT  
SQUARE ROOT= 2.0                      LOG= 1.386

go

YOU WILL NOW BE ASKED QUESTIONS CONCERNING  
WHAT YOU HAVE JUST LEARNED :

1.) IS THE FOLLOWING STATEMENT LEGAL  
REPLY : YES , OR NO.

```
LET Z1=4*SQR(3.0 * EXP(9.3))
```

yes

2. IS THE FOLLOWING SEQUENCE OF PROGRAM STATEMENTS LEGAL  
REPLY: YES OR NO.

```
LET B=-9
LET X=SQR(B)
...
...
...
```

no

NO , THE SQUARE ROOT OF A NEGATIVE NUMBER IS AN  
UNDEFINED OPERATION . IN THE NEXT LESSON YOU WILL BE SHOW A  
BASIC STATEMENT FOR TESTING AND BRANCHING TO ANOTHER SEGMENT  
OF THE PROGRAM IF THE TEST IS TRUE . FOR EXAMPLE THE ABOVE  
PROGRAM SEQUENCE MIGHT BE ALTERED AS FOLLOWS :

```
LET B=-9
IF B LT 0 THEN 100
50 LET X=SQR(X)
...
```





```

      ...
100 REM  NEGATIVE ARGUMENT
LET B=ABS(B)
GO TO 50
      ...
      ...

```

THIS PROGRAM SEQUENCE TESTS FOR A NEGATIVE ARGUMENT , IF TRUE IT BRANCHES TO STATEMENT NUMBER 100 , MAKES THE ARGUMENT POSITIVE AND BRANCHES BACK TO STATEMENT 50 TO COMPLETE THE PROGRAM .

go

### C. SUMMARY

WITH THE LET STATEMENT AND BUILT-IN FUNCTIONS , PLUS THE PREVIOUS BASIC STATEMENTS ( REM , READ , DATA , PRINT ) , YOU ARE FAST GAINING AN EFFECTIVE REPERTOIRE FOR PROGRAMMING USE . IN THE NEXT LESSON YOU WILL LEARN HOW TO SET UP LOOPS IN A PROGRAM SO THAT THE MAIN BODY OF A PROGRAM MAY BE EXECUTED AS OFTEN AS DESIRED . AS YOU ARE DOING YOUR REVIEW PROBLEMS , THINK ABOUT HOW YOU COULD SET UP A LOOP TO READ IN ANY AMOUNT OF DATA , PROCESS IT AND THEN HALT FOR SOME TEST CONDITION .

go

THE FOLLOWING REVIEW PROBLEMS WILL EXERCISE YOUR PROGRAMMING SKILLS TO DATE :

1.) WRITE A PROGRAM TO COMPUTE THE PRESENT WORTH OF AN INVESTMENT FOR SOME NUMBER OF YEARS HENCE . THE FORMULA IS :

$$P = S(1/((1+I)^N))$$

WHERE P IS THE PRESENT WORTH OF A PAYMENT S IN N YEARS HENCE AT AN INTEREST RATE OF I . FOR DATA USE I=.08 , S=5000 , N=20 .

2.) WRITE A PROGRAM TO FIND SIDES A , AND C OF A TRIANGLE USING THE LAW OF SINES FORMULA :

$$A/\sin(A) = B/\sin(B) = C/\sin(C)$$

WHERE THE NUMERATOR IS THE SIDE AND THE DENOMIATOR IS THE SINE OF THE ANGLE , THE CONVERSION FACTOR FROM DEGREES TO RADIAN IS:

$$1 \text{ DEGREE} = (3.1416/180) \text{ RADIAN} . \text{ THE DATA FOR THE PROGRAM IS :}$$

SIDE B=34.91 IN.      ANGLE A=98.71 DEG.  
 ANGLE B=49.97 DEG.      ANGLE C=31.32 DEG.

IF YOU WISH TO RUN THESE PROGRAMS NOW , THEN REPLY : YES  
 OTHERWISE REPLY : NO.

no



THE ANSWERS TO THE PROBLEMS WILL NOW BE GIVEN SO  
THAT YOU MAY CHECK YOUR RESULTS LATER :

1.) \$1072.74

2.) SIDE A=45.06 IN. AND SIDE C=23.69 IN.

DO YOU WANT TO TERMINATE YOUR INSTRUCTION SESSION ?  
(REPLY : YES OR NO )

yes

R; T=4.84/16.96 14.03.44



EXECUTION BEGINS...

\*\*\* LESSON 4 \*\*\*

THIS INSTRUCTION SEQUENCE WILL COVER BRANCHES . AS YOU HAVE SEEN FROM PREVIOUS PROGRAMS , A PROGRAM'S EXECUTION USUALLY TAKES A DIRECT ROUTE FROM THE FIRST TO THE LAST STATEMENT . THE CONDITION THAT ALLOWS DETOURS TO OCCUR IN PROGRAMS IS CALLED BRANCHING . THERE ARE TWO TYPES OF BRANCHING : UNCONDITIONAL AND CONDITIONAL .

go

A. UNCONDITIONAL BRANCHES

AN UNCONDITIONAL BRANCH IS AN IMPERATIVE TRANSFER OF CONTROL FROM ONE POINT IN A PROGRAM TO ANOTHER . THERE ARE TWO FORMS OF THE UNCONDITIONAL BRANCH : THE 'GO TO' AND THE 'COMPUTED GO TO'

1.) GO TO << STATEMENT NUMBER >>

THIS COMMAND TRANSFERS PROGRAM CONTROL DIRECTLY TO THE STATEMENT NUMBER AND CONTINUES EXECUTION FROM THAT POINT . THE 'GO TO' IS USED FOR FORMING LOOPS IN A PROGRAM .

go

A SAMPLE LOOP FOLLOWS :

```
REM PROGRAM TO COMPUTE PRESENT WORTH
REM P=INVESTMENT,S=PRINCIPAL,I=INTEREST RATE,N=NR.YEARS
PRINT 'INVESTMENT','PRINCIPAL','INTEREST','NR.YEARS'
10 READ S,I,N
LET P=S*(1/((1+I)**N))
PRINT P,S,I,N
GO TO 10
DATA 5000,.08,20,5000,.08,10,5000,.06,10
END
```

PRODUCES THE OUTPUT :

| INVESTMENT | PRINCIPAL | INTEREST | NR.YEARS |
|------------|-----------|----------|----------|
| 1072.74    | 5000      | .08      | 20       |
| 2315.97    | 5000      | .08      | 10       |
| 2791.99    | 5000      | .06      | 10       |

\*\*\* ERROR , YOU TRIED TO READ MORE NUMERIC DATA THAN YOU PUT IN \*\*\*



go

THE ERROR OCCURS BECAUSE YOU RUN OUT OF DATA DURING THE EXECUTION OF THE LOOP SET-UP BY THE UNCONDITIONAL TRANSFER . IF THE READ STATEMENT WERE NOT IN THE LOOP TO CAUSE THE PROGRAM TO STOP . THEN YOU WOULD BE IN AN 'INFINITE LOOP' , A CONDITION IN WHICH THERE IS NO WAY TO STOP . YOU MUST ALWAYS CHECK FOR THE 'INFINITE LOOP' CONDITION BY MAKING SURE THAT YOUR PROGRAM HAS AN EXIT .

go

2.) ON << EXPRESSION >> GO TO << STATEMENT NUMBER ,...., STATEMENT NUMBER >>

THIS SPECIAL FORM OF THE 'GO TO' COMMAND , IS CALLED THE 'COMPUTED GO TO' , THE EXPRESSION IN THE FORM OF THE STATEMENT MUST EVALUATE TO AN INTEGER NUMBER BETWEEN 1-->9999 . IF IT IS NOT AN AN INTEGER , OR OUTSIDE THIS RANGE AN ERROR WILL OCCUR .

WHEN THE 'COMPUTED GO TO' IS EXECUTED THE EXPRESSION IS EVALUATED , AND PROGRAM CONTROL TRANSFERS TO THE N-TH STATEMENT NUMBER , WHERE N-TH REPRESENTS THE VALUE OF THE EXPRESSION . FOR EXAMPLE :

```
LET I=3
ON I GO TO 100,33,475,9999
```

go

EXECUTION OF THE 'COMPUTED GO TO' WOULD CAUSE PROGRAM CONTROL TO TRANSFER UNCONDITIONALLY TO STATEMENT NUMBER 475 . YOU MUST BE CAREFUL WHEN USING THE 'COMPUTED GO TO' NOT ONLY BECAUSE OF INFINITE LOOPS ; BUT BECAUSE THE EXPRESSION MUST BE AN INTEGER BETWEEN 1-->9999 , AND THERE MUST BE A STATEMENT NUMBER FOR 'ALL' POSSIBLE VALUES OF THE EXPRESSION .

go

### B. CONDITIONAL BRANCHING

THE CONDITIONAL BRANCH TRANSFERS CONTROL ONLY IF CERTAIN RELATIONS ARE TRUE . IF THE THE TEST OF RELATIONS IS TRUE THEN TRANSFER OF CONTROL OCCURS ; OTHERWISE PROGRAM CONTROL CONTINUES WITH THE NEXT STATEMENT . THE FORM OF THE CONDITIONAL BRANCH IS:

IF << EXPRESSION >> << RELATION >> << EXPRESSION >> THEN << STATEMENT NUMBER >>

NOTE THAT ALPHA VARIABLES ARE NOT ALLOWED AS AN EXPRESSION IN A CONDITIONAL BRANCH .





go

THE RELATIONS ARE :

| SYMBOLS | EXAMPLE | MEANING                      |
|---------|---------|------------------------------|
| GT      | A GT B  | A GREATER THAN B             |
| GE      | A GE B  | A GREATER THAN OR EQUAL TO B |
| LT      | A LT B  | A LESS THAN B                |
| LE      | A LE B  | A LESS THAN OR EQUAL TO B    |
| NE      | A NE B  | A NOT EQUAL TO B             |
| =       | A = B   | A EQUAL B                    |

go

WHEN THE CONDITIONAL STATEMENT IS EXECUTED THE  
( EXPRESSION RELATION EXPRESSION ) IS TESTED , AND IF THE RELATION  
IS TRUE , THEN CONTROL IS TRANSFERRED TO THE STATEMENT NUMBER.  
OTHERWISE PROGRAM CONTROL CONTINUES TO THE NEXT SEQUENTIAL  
STATEMENT , FOR EXAMPLE :

```
1 READ A
...
...
IF A LT 0 THEN 90
LET X=SQR(A)
...
...
90 PRINT 'ILLEGAL ARGUMENT',A
GO TO 1
...
```

THE ONLY TIME THAT THE CONDITIONAL BRANCH IS EXECUTED IS WHEN  
A IS LESS THAN 0 .

go

YOU WILL NOW BE ASKED SOME QUESTIONS ABOUT BRANCHING :

1.) ARE THE FOLLOWING STATEMENTS CORRECT  
REPLY: YES OR NO)

GO TO END

no

IF X1 LT A\$ THEN 10

no



```
IF(X**3-4) GE 10 THEN GO TO 100
```

yes

YOUR ANSWER IS INCORRECT . THE ONLY THING ALLOWED  
AFTER THEN IS A STATEMENT NUMBER .

```
LET Y=6,  
ON Y GO TO 1,3,5,7,999
```

no

2.) CONSIDER THIS PROGRAM SEGMENT , ANY ERRORS (REPLY: YES OR NO).

```
1 READ A,B  
3 LET Z=A*B  
IF Z LT 0 GO TO 999  
...  
...  
GO TO 3  
DATA 3,5,7,10  
999 END
```

yes

THERE IS AN 'INFINITE' LOOP IN THIS PROGRAM . IT  
COULD BE CORRECTED BY CHANGING THE GO TO 3 , TO GO TO 1 .  
WHEN YOU WRITE PROGRAMS HAVING BRANCHES OR LOOPS YOU MUST ALWAYS  
CONSIDER HOW THE PROGRAM WILL STOP. YOU HAVE OBSERVED THAT  
AN 'INFINITE LOOP' CAN BE STOPPED BY HAVING A READ STATEMENT  
IN THE LOOP AND JUST RUN OUT OF DATA . HOWEVER, ENDING A PROGRAM  
ON AN ERROR IS AN INELIGANT METHOD . THE MOST COMMON METHODS  
USE THE 'GO TO' AND ' IF/THEN' COMMANDS TO CONTROL PROGRAM  
LOOPS AND ARE AS FOLLOWS :

go

1. COUNT AND TEST METHOD , IN WHICH A COUNTER  
IS INCREMENTED IN THE LOOP , AND WHEN THE COUNTER REACHES A  
CERTAIN VALUE , BRANCH OUT OF THE LOOP . EXAMPLE PROGRAM

```
REM COUNT AND TEST METHOD  
REM N IS COUNTER , INITIALIZED TO 0 , AND COUNTS FROM 1-->10
```



```

LET N=Z=0
READ X
10 LET Z=Z+X
LET N=N+1
IF N GT 10 THEN 100
GO TO 10
100 PRINT'SUM=',Z
DATA 10
END

```

REPLY WITH VALUE OF Z .

11

YOUR ANSWER IS INCORRECT . THE ONLY WAY TO BE SURE OF VALUES IN A LOOP IS TO SET UP A TABLE OF THE VARIABLES AND KEEP TRACK OF THERE VALUES IN THE LOOP :

| X  | N   | Z   |
|----|-----|-----|
| 10 | 0   | 0   |
|    | 1   | 10  |
|    | 2   | 20  |
|    | 3   | 30  |
|    | ... |     |
|    | 9   | 90  |
|    | 10  | 100 |
|    | 11  | 110 |

THUS Z=110

2. READ AND TEST METHOD , IN WHICH A VALUE IS READ IN AND TESTED FOR THE END OF LOOP CONDITION .

```

REM READ AND TEST DEMONSTRATION
REM 9999 IS END VALUE
LET Z=0
1 READ X
IF X=9999 THEN 9999
LET Z=Z+X
GO TO 1
DATA 1,2,3,4,5,6,9999
9999 PRINT 'SUM=',Z
END

```

REPLY WITH VALUE OF Z

21



### C. SUMMARY

YOU HAVE SEEN HOW THE UNCONDITIONAL BRANCHES ('GO TO' AND 'COMPUTED GO TO') TRANSFER PROGRAM CONTROL , AND HOW THE CONDITIONAL BRANCH ('IF/THEN') TESTS FOR TRANSFER OF PROGRAM CONTROL ; AND YOU HAVE OBSERVED THE CONTROL OF LOOPS SO THAT A PROGRAM SEGMENT MAY BE REPEATED UNTIL A SPECIFIED CONDITION IS MET. IN THE NEXT LESSON YOU WILL LEARN A BASIC STATEMENT TO CONTROL A LOOP BY THE INCREMENT AND TEST METHOD .

THE FOLLOWING PROBLEMS WILL TEST YOUR NEW SKILLS :

go

1.) WRITE A PROGRAM TO COUNT THE NUMBERS BETWEEN 50 AND 60 , AND ALSO PRINT THEM OUT . THE INPUT DATA IS 10,50,35,75,62,60,54,54 .

2.) WRITE A PROGRAM TO COMPUTE THE PRESENT WORTH OF AN INVESTMENT FOR SOME YEARS HENCE , AT VARIOUS INTEREST RATES . THE FORMULA IS:  
$$P = S / ((1 + I)^N)$$
 WHERE P=PRESENT WORTH, S=PRINCIPAL, I=INTEREST, AND N=NR.OF YEARS.  
FOR DATA USE S=5000, AND I=.04-->.8, IN INCREMENTS OF .01 , AND N=20

IF YOU DESIRE TO EXECUTE THESE PROGRAMS NOW  
REPLY: YES ; OTHERWISE REPLY : NO

no

DO YOU WANT TO TERMINATE YOUR INSTRUCTION SESSION ?  
(REPLY : YES OR NO )

no

IF YOU WANT TO EXECUTE PROGRAMS AT THIS TIME  
THEN REPLY : YES ; OTHERWISE REPLY : NO AND THE INSTRUCTION  
WILL CONTINUE .

no

REPLY WITH THE NUMBER OF THE LESSON YOU WISH TO COVER





EXECUTION BEGINS...

\*\*\* LESSON 5 \*\*\*  
THIS LESSON WILL INTRODUCE YOU TO ITERATION , SUBSCRIPTED  
VARIABLES , AND LISTS (VECTORS) AND TABLES (MATRICES).

A. ITERATION (LOOPING)

IN THE LAST LESSON YOU WERE SHOWN HOW TO USE CONDITIONAL AND  
UNCONDITIONAL BRANCHES TO CONTROL LOOPING . THE COUNT AND TEST  
ITERATIVE LOOP OCCURS SO FREQUENTLY THAT AN ABBREVIATED BASIC  
STATEMENT HAS BEEN DEvised TO CONTROL LOOPING . THE ITERATIVE  
LOOP HAS THE FOLLOWING FORM :

```
FOR << SIMPLE VARIABLE >> = << EXPRESSION >> TO << EXPRESSION >>  
    STEP << EXPRESSION >>  
    ...  
    ...  
    ...  
NEXT << SIMPLE VARIABLE >>
```

go

FOR EXAMPLE CONSIDER THIS PROGRAM SEGMENT :

```
FOR I=1 TO 10 STEP 2  
  LET X=X+I  
NEXT I
```

THE SIMPLE VARIABLE FOLLOWING 'FOR' IS THE LOOP INDEX . WHEN  
THE FOR/NEXT PAIR IS EXECUTED , THE LOOP INDEX IS GIVEN THE  
VALUE (INITIALIZED) OF THE FIRST EXPRESSION (I=1 IN EXAMPLE).  
THIS INDEX IS THEN TESTED TO DETERMINE WHETHER IT IS 'GREATER THAN'  
THE SECOND EXPRESSION AFTER 'TO' (10 IN EXAMPLE). IF IT IS  
GREATER, CONTROL IS TRANSFERRED TO THE STATEMENT FOLLOWING  
'NEXT'. OTHERWISE THE REMAINING STATEMENTS WITHIN THE LOOP  
(FOR/NEXT) ARE EXECUTED SEQUENTIALLY UNTIL THE 'NEXT' STATEMENT  
IS REACHED.

go

WHEN THE 'NEXT' STATEMENT IS REACHED , THE LOOP  
INDEX IS INCREASED (INCREMENTED) BY THE AMOUNT OF THE EXPRESSION  
FOLLOWING 'STEP', AND CONTROL IS TRANSFERRED BACK TO THE 'FOR'  
STATEMENT WHERE THE LOOP CONTINUES UNTIL THE INDEX VALUE IS



GREATER THAN THE FINAL VALUE . FOR EXAMPLE :

```
REM DEMO LOOP. SUM THE NUMBERS FROM 1-->10.
LET C=0
FOR I=1 TO 10 STEP 1
LET C=C+1
NEXT I
PRINT'SUM=',C
END
```

go

YOU WILL NOTICE THAT THE SIMPLE VARIABLE FOLLOWING 'NEXT' IS THE SAME AS THE SIMPLE VARIABLE FOLLOWING 'FOR', AND THAT THE 'NEXT' STATEMENT MARKS THE END OF THE LOOP . BECAUSE THE INCREMENT VALUE OF A LOOP IS COMMONLY ONE(1) , THE 'STEP' MODIFIER AND ITS EXPRESSION MAY BE OMITTED , AND THE INCREMENT VALUE WILL BE ASSUMED TO BE ONE(+1). FOR EXAMPLE THE ABOVE 'FOR' STATEMENT COULD BE WRITTEN :

```
FOR I=1 TO 10
```

THE INCREMENT VALUE AFTER 'STEP' MAY BE POSITIVE OR NEGATIVE ALLOWING THE FLEXIBILITY OF LOOPING FORWARD OR BACKWARD . FOR A NEGATIVE 'STEP' VALUE THE TEST BECOMES 'LESS THAN'. FOR EXAMPLE THE FOLLOWING 'FOR' STATEMENTS ARE EQUIVALENT :

```
FOR I=1 TO 10
FOR I=10 TO 1 STEP -1
```

go

ANOTHER USEFUL TECHNIQUE OF LOOPING IS 'NESTING' NESTING REFERS TO PLACING ONE LOOP INSIDE ANOTHER LOOP . THE INNER LOOP 'SPINS' AROUND AS MANY TIMES AS THE OUTER LOOP IS INCREMENTED . FOR EXAMPLE CONSIDER THIS PROGRAM SEGMENT

```
FOR I=1 TO 10
FOR J=1 TO 20 STEP 2
...
...
NEXT J
NEXT I
```

IN THIS EXAMPLE THE OUTSIDE LOOP(I) IS



REPEATED 10 TIMES, AND THE INNER LOOP(J) WOULD BE REPEATED 20 TIMES FOR EACH INCREMENT OF THE OUTSIDE LOOP, OR 200 REPETITIONS. LOOPS MAY BE NESTED UP TO A MAXIMUM OF 20 ; HOWEVER , THEY CANNOT OVERLAP . THE INNERMOST LOOP MUST BE CLOSED WITH ITS 'NEXT' STATEMENT BEFORE ENCOUNTERING THE NEXT OUTER LOOP'S 'NEXT' STATEMENT . FOR EXAMPLE :

```
FOR X=10 TO 1 STEP -1
FOR Y=3 TO 5
FOR Z=-3 TO -5 STEP -1
...
NEXT Z
NEXT Y
NEXT X
```

go

WITHIN A FOR/NEXT LOOP CONDITIONAL AND UNCONDITIONAL BRANCHES MAY BE USED TO TRANSFER CONTROL OUT OF A LOOP OR WITHIN LIMITS OF THE SAME LOOP . HOWEVER , IT IS NOT POSSIBLE TO BRANCH INTO THE MIDDLE OF A FOR/NEXT LOOP BECAUSE LOGIC PROBLEMS OCCUR AND AN ERROR WILL RESULT . AN ADDITIONAL ITEM TO BE CAREFUL ABOUT IS USING THE INDEX VARIABLE OF THE FOR/NEXT LOOP IN COMPUTATIONS . IF YOU ALTER THE VALUE OF THE LOOP INDEX YOU WILL EFFECT THE ACTION OF THE LOOP . FOR EXAMPLE :

```
FOR I=1 TO 10
LET I=I+10
NEXT I
PRINT I
END
```

REPLY WITH VALUE OF I THAT IS PRINTED

11

YOU WILL NOW BE ASKED SOME QUESTIONS ABOUT WHAT YOU HAVE JUST LEARNED .

```
1.) SAMPLE PROGRAM
LET S=0
FOR K=-5 TO 5
LET S=S+K
NEXT K
PRINT'SUM=',S
END
```

REPLY WITH VALUE OF S .

0

```
2.) SAMPLE PROGRAM
FOR I=100 TO 1 STEP 2
LET Z=I**2
LET Y= Z+10
```



```

NEXT I
PRINT I
END

```

REPLY WITH VALUE OF I

1

YOUR ANSWER IS INCORRECT . THE INDEX VALUE OF THE LOOP(I=100) IS GREATER THAN THE FINAL VALUE(1) IN THE FIRST TEST . THUS I=100 . IF THE LOOP WERE DECREMENTED IN STEPS OF -2 THEN THE LOOP WOULD BE EXECUTED AND THEN I=0 .

```

3.) SAMPLE PROGRAM
LET T=0
FOR I=5 TO 1 STEP -1
FOR J=1 TO 5
IF I=J THEN 10
GO TO 15
10 LET T=T+1
PRINT I
15 NEXT J
NEXT I
PRINT T
END

```

REPLY WITH VALUE OF T .

5

## B. SUBSCRIPTED VARIABLES/LISTS AND TABLES

1. IN LESSON 1 YOU LEARNED THAT SIMPLE VARIABLES WERE A LETTER OR A LETTER FOLLOWED BY A DIGIT . SUBSCRIPTED VARIABLES CONSIST OF A LETTER FOLLOWED BY A SINGLE OR DOUBLE SUPSCRIPT IN PARENTHESIS THE SUBSCRIPTS MAY BE ANY LEGAL EXPRESSION THAT EVALUATES TO AN INTEGER VALUE . FOR EXAMPLE :

A(1) , B(3) , D(1,3) , Z(I,J) , X(3\*\*A) , D(A(1)) ARE LEGAL SUBSCRIPTED VARIABLE





BUT A1(1) , Z(1,2,3) ARE ILLEGAL SUBSCRIPTED VARIABLES.

A USE OF SUBSCRIPTED VARIABLES FOLLOWS :

go

2. A NUMERIC LIST , OR VECTOR , OR SINGLE DIMENSIONED ARRAY , IS A SET OF NUMERIC VALUES ARRANGED IN AN ORDERLY MANNER . FOR EXAMPLE , SUPPOSE YOU WOULD LIKE TO READ SOME NUMBERS INTO YOUR PROGRAM AND HAVE THESE NUMBERS AVAILABLE AND IDENTIFIED FOR USE AT A LATER TIME . YOU COULD ASSIGN SIMPLE VARIABLES TO EACH VALUE , BUT WHAT IF YOU HAD 100 NUMBERS ? IT IS EASIER TO THINK OF THE NUMBERS AS A LIST OF VALUES OR A VECTOR THE SIZE OF WHICH IS DETERMINED BY HOW MANY NUMBERS YOU HAVE . YOU THEN SIMPLY ASSIGN THE VALUES TO THE LIST .

go

THE FOLLOWING PROGRAM WILL ASSIGN 10 VALUES TO THE LIST 'A' WHICH CONTAINS 10 ELEMENTS :

```
DIM A(10)
FOR L=1 TO 10
  READ A(L)
NEXT L
DATA 1,3,7,5,6,9,4,2,14,4
END
```

THE LIST A , CONTAINING 10 ELEMENTS , IS REPRESENTED BY THE SUBSCRIPTED VARIABLE A(L) . READ A(L) IS IN A FOR/NEXT LOOP , AND WITHIN THE LOOP THE LIST ELEMENTS A(1)-->A(10) ARE ASSIGNED VALUES . NOW THE LIST A HOLDS THE 10 VALUES AND EACH VALUE IS IDENTIFIABLE . CONSIDER THE FOLLOWING PROBLEM WHICH SEARCHES A LIST TO FIND THE LARGES

go

```
DM N(10)
FOR I=1 TO 10
  LET N(I)=0
NEXT I
READ K
FOR J=1 TO K
  READ N(J)
NEXT J
LET L=N(1)
FOR J=2 TO K
  IF N(J) LT L THEN 10
  LET L=N(J)
```



```

10 NEXT J
PRINT 'LARGEST NUMBER=',L
DATA 5,3,5,9,10,6
END

```

REPLY WITH VALUE OF L

10

THE FIRST FOR/NEXT LOOP ZEROS OUT THE LIST .  
 IT IS A GOOD PRACTICE TO PUT SOME VALUES IN THE LISTS YOU USE  
 OTHERWISE THE COMPUTER MAY ASSIGN RANDOM VALUES . BY PUTTING  
 ZERO IN EACH ELEMENT OF THE LIST YOU ARE ASSURED THAT THE LIST  
 IS CLEANED UP BEFORE YOU USE IT.

go

3. IN ADDITION TO LISTS(VECTORS), BASIC ALLOWS  
 YOU THE ABILITY TO USE TABLES , OR MATRICES , OR TWO DIMENSIONAL  
 ARRAYS . THE SUBSCRIPTS OF A TABLE REPRESENT THE ROWS AND  
 COLUMNS OF THE TABLE. THE FIRST SUBSCRIPT IS THE ROW AND THE  
 SECOND SUBSCRIPT IS THE COLUMN . FOR EXAMPLE IN TABLE'D':  
 D(3,4) REFERS TO THE VALUE OF THE ELEMENT IN ROW 3 , COLUMN 4  
 OF TABLE'D'.

go

AS WITH A LIST , ANY ELEMENT OF A TABLE MAY BE  
 REFERENCED BY DEFINING THE PAIR OF SUBSCRIPTS AS DESIRED IN  
 ROW-COLUMN ORDER . IN A 3X3 TABLE(MATRIX) THE TABLE IS REFERENCED  
 AS FOLLOWS :

|     |     |     |
|-----|-----|-----|
| 1,1 | 1,2 | 1,3 |
| 2,1 | 2,2 | 2,3 |
| 3,1 | 3,2 | 3,3 |

WHERE THE FIRST SUBSCRIPT IDENTIFIES THE ROW AND THE SECOND THE COLUMN .  
 USING THIS REFERENCE SYSTEM CONSIDER HOW TO FILL THE FOLLOWING TABLE

|   |   |   |
|---|---|---|
| 1 | 2 | 3 |
| 4 | 5 | 6 |
| 7 | 8 | 9 |

go

THIS TABLE'X' COULD BE FILLED AS FOLLOWS :

```

DIMX(3,3)
FOR I=1 TO 3
FOR J=1 TO 3
READ X(I,J)
NEXT J
NEXT I
DATA 1,2,3,4,5,6,7,8,9

```



END

REPLY WITH VALUE OF X(2,3)

6

NO WITH TABLE 'X' ASSIGNED VALUES CONSIDER THIS  
PROGRAM (ASSUMING TABLE 'X' HAS BEEN FILLED BY THE ABOVE PROGRAM)

```
LET S=0
FOR I=1 TO 3
FOR J=1 TO 3
IF I NE J THEN 5
LET S=S+X(I,J)
5 NEXT J
NEXT I
PRINT'SUM=',S
END
```

REPLY WITH VALUE OF S

15

YOU HAVE SEEN HOW SUBSCRIPTED VARIABLES ARE  
USED TO SET UP LISTS (VECTORS) AND TABLES (MATRICES). HOWEVER TO  
USE LISTS AND TABLES IN A PROGRAM, YOU MUST DIMENSION THE  
MAXIMUM SIZE OF YOUR LIST OR TABLE SO THAT ENOUGH SPACE WILL  
BE ALLOCATED IN THE COMPUTER MEMORY. THIS DIMENSIONING IS DONE  
WITH THE DIM STATEMENT.

80

### C. DIM STATEMENT

THE DIM STATEMENT TELLS THE COMPUTER THE MAXIMUM SIZE OF VECTORS  
AND TABLES THAT WILL BE USED IN YOUR PROGRAM. THE DIM STATEMENT  
MUST APPEAR IN THE PROGRAM BEFORE 'ANY' REFERENCE IS MADE  
TO THE LIST OR TABLE. IN GENERAL PRACTICE THE DIM STATEMENT  
IS USUALLY THE FIRST STATEMENT IN THE PROGRAM. THE FORM OF THE  
DIM STATEMENT IS :

```
DIM <<LIST VARIABLE>> ( <<SIZE>> )
```

```
DIM <<TABLE VARIABLE>> ( <<SIZE,SIZE>> )
```

80



THE LIST AND TABLE VARIABLES ARE SUBSCRIPTED VARIABLES AS DEFINED EARLIER . THE SIZE IS AN UNSIGNED INTEGER IN PARENTHESIS WHICH DENOTES THE MAXIMUM SIZE OF THE LIST OR TABLE THE DIM STATEMENT MAY CONTAIN A NUMBER OF LISTS OR TABLES WITH THEIR SIZES SEPARATED BY COMMAS , FOR EXAMPLE :

```
DIM A(6),X(10,3) ,Z(14,21)
```

go

IF YOU TRY TO REFERENCE AN ELEMENT IN A LIST OR TABLE BEYOND THE MAXIMUM SIZE IN THE DIM STATEMENT YOU WILL GET AN ERROR . THE MAXIMUM SIZE LIST(VECTOR) OR TABLE(MATRIX) ALLOWED BY THE CAI-BASIC COMPILER IN ANY ONE PROGRAM IS A TOTAL OF 1600 COMPUTER MEMORY SPACES .

YOU WILL NOW BE ASKED SOME QUESTIONS :

1.) ARE THE FOLLOWING DIM STATEMENTS CORRECT,REPLY : YES OR NO

```
DIM A1(10)
```

yes

THE STATEMENT IS INCORRECT . SUBSCRIPTED VARIABLES ARE A 'SINGLE' LETTER FOLLOWED BY ONE OR TWO SUBSCRIPTS IN PARENTHESIS.

```
DIM X(500),B(1,10),C(5,5,5)
```

no

#### D. SUMMARY

IN THIS LESSON YOU HAVE LEARNED HOW TO USE THE FOR/NEXT STATEMENT AND HOW TO MANIPULATE LISTS(VECTORS) AND TABLES(MATRICES) BY USING SUBSCRIPTS VARIABLES AND THE DIM STATEMENT . YOU NOW HAVE ALL THE TOOLS TO BEGIN WRITING SOPHISTICATED PROGRAMS ; AND AS YOU WRITE MORE COMPLICATED PROGRAMS , YOU WILL FIND A NEED FOR SUBROUTINES . SUBROUTINES ARE COMMONLY USED PROGRAM SEGMENTS THAT ARE USED OVER AGAIN IN OTHER PARTS OF YOUR PROGRAM SUBROUTINES ALLOW YOU TO BRANCH TO THE COMMONLY USED SEGMENT AND THEN RETURN TO WHERE YOU WERE AND CONTINUE EXECUTING . THE SUBROUTINE CAN BE 'CALLED' FROM ANYWHERE IN YOUR PROGRAM . THE GOSUB STATEMENT ALLOWS SUBROUTINES IN BASIC , AND YOU WILL BE





INTRODUCED TO IT IN THE NEXT LESSON .

YOU WILL NOW BE GIVEN TWO OPTIONAL PROGRAMMING  
PROBLEMS TO EXERCISE YOUR NEW TOOLS .

1.) WRITE A PROGRAM TO REVERSE THE NUMBERS IN A 10-ELEMENT LIST 'X'  
IN OTHER WORDS INTERCHANGE X(1) WITH X(10), X(2) WITH X(9), ETC.  
READ IN TEN VALUES AND TEST YOUR PROGRAM BY PRINTING THE LIST  
BEFORE AND AFTER .

2.) WRITE A PROGRAM TO ARRANGE THE FOLLOWING LIST IN  
DECENDING ORDER : 10,30,5,15,40 . ONE METHOD OF APPROACHING THIS  
IS TO CHECK THE FIRST ELEMENT OF THE LIST AGAINST THE SECOND .  
IF THE FIRST IS NOT LARGER THEN EXCHANGE THE TWO , OTHERWISE  
GO AND COMPARE THE NEXT TWO IN THE LIST. THIS PROCESS IS REPEATED  
UNTIL THERE ARE NO MORE EXCHANGES TO BE MADE . A COUNT OF  
THE EXCHANGES CAN BE MADE , AND WHEN THE COUNT EQUALS 0 THE LIST IS IN ORDER.

go

IF YOU WANT TO EXECUTE THESE PROBLEMS REPLY : YES , OTHERWISE NO ,

no

IF YOU WOULD LIKE TO SEE A SOLUTION TO THE PROBLEMS REPLY : YES , OTHERWISE NO

no

DO YOU WANT TO TERMINATE YOUR INSTRUCTION SESSION ?  
(REPLY : YES OR NO )

no

IF YOU WANT TO EXECUTE PROGRAMS AT THIS TIME  
THEN REPLY : YES ; OTHERWISE REPLY : NO AND THE INSTRUCTION  
WILL CONTINUE .

no

REPLY WITH THE NUMBER OF THE LESSON YOU WISH TO COVER



EXECUTION BEGINS...

\*\*\* LESSON 6 \*\*\*  
THIS LESSON WILL INTRODUCE YOU TO SUBROUTINES AND THE PROGRAMMING  
TECHNIQUE CALLED 'RECURSION'.

A. SUBROUTINES

IN MANY PROGRAMS A BLOCK OF STATEMENTS MAY BE NEEDED ON MORE  
THAN ONE OCCASION . SINCE THE REPETITION OF A BLOCK OF STATEMENTS  
IS BURDENSOME , A TECHNIQUE TO ECONOMIZE ON CODING INSTRUCTIONS  
CALLED A SUBROUTINE , IS PRESENTED. THE FORM OF THE SUBROUTINE IS :

```
GOSUB << STATEMENT NUMBER >>
...
...
...
RETURN
```

go

EXECUTION OF THE GOSUB STATEMENT CAUSES THE  
COMPUTER TO TRANSFER CONTROL TO THE STATEMENT NUMBER AFTER 'GOSUB'  
WHEN CONTROL IS TRANSFERRED , STATEMENTS ARE EXECUTED SEQUENTIALLY  
UNTIL A 'RETURN' IS ENCOUNTERED . AT THAT TIME , CONTROL  
IS RETURNED TO THE NEXT STATEMENT FOLLOWING THE 'GOSUB'  
STATEMENT WHICH 'CALLED' THE SUBROUTINE . FOR EXAMPLE CONSIDER  
THE PROGRAM SEGMENT :

go

```
...
PRINT A,B,C,D,E
GOSUB 500
FOR I=1 TO E
...
...
...
500 REM SUBROUTINE
LET X=A+B
LET Z=B+C
LET R=SQR(D)
RETURN
...
```

go



THE SUBROUTINE CONSISTS OF A SEQUENCE OF BASIC STATEMENTS ENDING WITH A 'RETURN' STATEMENT . THE SUBROUTINE MAY ONLY BE ENTERED FROM A 'GOSUB' STATEMENT , AND WILL ONLY RETURN TO ITS PROPER PLACE AFTER ENCOUNTERING A 'RETURN' STATEMENT .

ALL THE VARIABLES OF THE MAIN PROGRAM ARE AVAILABLE (PASSED) TO THE SUBROUTINE , AND VICE VERSA . THE VARIABLES IN THE SUBROUTINE MUST BE CHOSEN CAREFULLY SO AS NOT TO ACCIDENTLY CONFLICT OR ALTER VARIABLES IN THE MAIN PROGRAM .

go

THE FOLLOWING PROGRAM TO COMPUTE THE GREATEST COMMON DENOMINATOR(GCD) OF THREE NUMBERS BY EUCLID'S ALGORITHM WILL DEMONSTRATE A USE OF SUBROUTINES :

```

      REM FIND GCD OF A,B,C
      PRINT 'A','B','C','GCD'
20  READ A,B,C
      IF C=9999 THEN 9999
      LET X=A
      LET Y=B
      REM G=GCD OF A,B
      GOSUB 200
      LET X=G
      LET Y=C
      REM G=GCD OF G,C
      GOSUB 200
      PRINT A,B,C,G
      GO TO 20
      200 LET Q=INT(X/Y)
          LET R=X-Q*Y
          IF R=0 THEN 300
          LET X=Y
          LET Y=R
          GO TO 200
      300 LET G=Y
      RETURN
      DATA 60,90,120
      DATA 38456,64872,98765
      DATA 0,0,9999
9999 END

```

OUTPUT PRODUCED :

| A     | B     | C     | GCD |
|-------|-------|-------|-----|
| 60    | 90    | 120   | 30  |
| 38456 | 64872 | 98765 | 1   |

go



GOSUB STATEMENTS MAY BE NESTED TO LOGICALLY CALL ANOTHER GOSUB , SIMILAR TO FOR/NEXT NESTED LOOPS . YOU MUST ONCE MORE BE CAREFUL ABOUT THE STATUS OF VARIABLES BECAUSE VARIABLES WILL BE PASSED FROM ONE SUBROUTINE TO ANOTHER .  
A SUBROUTINE THAT IS NESTED SO THAT IT CALLS ITSELF IS CALLED RECURSION OF SUBROUTINES OR RECURSION .

80

#### B. RECURSION

RECURSION IS A PROGRAMMING TECHNIQUE IN WHICH SUBROUTINES CAN CALL THEMSELVES .FOR EXAMPLE CONSIDER THIS PROGRAM WHICH FINDS THE FACTORIAL OF A NUMBER USING ITERATIVE METHODS AND THEN USING RECURSION :

```

      REM FACTORIAL : F(N)=1*2*3*....*(N-1)*N
      PRINT 'ITERATIVE SOLUTION'
      PRINT 'N', 'F(N)'
10  READ N
      IF N GT 0 THEN 20
      GO TO 50
20  PRINT N,
      GOSUB 100
      PRINT F
      GO TO 10
      100 REM ITERATIVE SOLUTION
          LET F=1
          FOR I=1 TO N
              LET F=F*I
          NEXT I
          REM RETURN F=FACTORIAL(N)
      RETURN
50  REM FACTORIAL : F(N)= IF N=0 THEN F(N)=1
      REM OTHERWISE , F(N)=N*F(N-1)
      RESTORE
      PRINT 'RESURSIVE SOLUTION'
60  READ N
      IF N GT 0 THEN 70
      GO TO 9999
70  PRINT N,
      GO SUB 200
      PRINT F
      GO TO 60
      200 REM RECURSIVE SOLUTION
          IF N GT 0 THEN 210
          LET F=1
          RETURN
      210 LET N=N-1
          REM RECURSIVE CALL

```





```

        GOSUB 200
        LET N=N+1
        LET F=F*N
    RETURN
    DATA 2,5,8,6,-1
999 END

```

go

IN THE ITERATIVE METHOD THE FACTORIAL FUNCTION ,  $F(N)$  , WAS MULTIPLIED BY ITSELF IN A LOOP FROM  $I=1 \rightarrow N$  . IN THE RECURSIVE SOLUTION THE FACTORIAL FUNCTION ,  $F(N)$  , IS DEFINED IN TERMS OF ITS FINAL VALUE . WHEN  $N=0$  ,  $F(N)=1$  ; OTHERWISE  $F(N)=N*(\text{FACTORIAL OF } N-1)$  .

TO KEEP TRACK OF WHERE THE SUBROUTINE CALLS RETURN IN RECURSION IT HELPS TO EVISIONING A LAST-IN-FIRST-OUT (LIFO) STACK CONTAINING THE ADDRESS OF THE GOSUB STATEMENTS. EVERY TIME A GOSUB IS ENCOUNTERED , PUT ITS ADDRESS ON TOP OF THE STACK (PUSHING DOWN ANYTHING PREVIOUSLY ON THE STACK). THEN EVERY TIME A RETURN IS ENCOUNTERED , RETURN TO THE TOP ADDRESS ON THE STACK (AND POP UP THE NEXT ADDRESS ON THE STACK).

go

TO FULLY UNDERSTAND THE CONCEPT OF RECURSION YOU SHOULD STEP THROUGH THE FACTORIAL PROBLEM BY HAND USING THE HELP OF THE STACK TO SEE HOW RECURSION WORKS . IF YOU UNDERSTAND THE CONCEPT OF RECURSION YOU ARE READY TO SOLVE THIS PROBLEM :

```

    LET X=1
    GOSUB 100
    PRINT X
    GO TO 9999
    100 LET X=X+1
        IF X GT 3 THEN 150
        GOSUB 100
    150 LET X=X+1
    RETURN
9999 END

```

REPLY WITH VALUE OF X

5



YOUR ANSWER IS INCORRECT . CONSIDER THE FOLLOWING  
TABLE OF VALUES FOR X , AND ITEMS IN THE STACK(1-FIRST GOSUB, 2-SECOND GOSUB)

|     |       |
|-----|-------|
| X=1 | STACK |
|     | 1     |
| X=2 | STACK |
|     | 2     |
|     | 1     |
| X=3 | STACK |
|     | 2     |
|     | 2     |
|     | 1     |
| X=4 |       |
| X=5 | STACK |
|     | 2     |
|     | 1     |
| X=6 | STACK |
|     | 1     |
| X=7 |       |

YOU SHOULD STEP THROUGH THIS  
EXAMPLE UNTIL YOU UNDERSTAND THE RECURSION TECHNIQUE . HOWEVER,  
KNOWING HOW TO USE RECURSION IS NOT A REQUIREMENT FOR KNOWING  
HOW TO USE BASIC , IT IS ONLY A CLASSIC PROGRAMMING TECHNIQUE .

#### C. SUMMARY

SUBROUTINES AND RECURSION ARE A VALUABLE ADDITION TO YOUR REPERTOIR  
OF BASIC STATEMENTS . THEY SAVE BOTH PROGRAMMER TIME AND  
COMPUTER STORAGE SPACE . THE GOSUB/RETURN COMMAND AND THE OTHER  
TWELVE BASIC STATEMENTS THAT YOU HAVE ALREADY LEARNED AND USED  
FORM THE BASIC LANGUAGE .

THE FACILITY THAT YOU GAIN IN PROGRAMMING BY USING THE BASIC  
LANGUAGE WILL OPEN UPON HOW OFTEN YOU EXERCISE YOUR SKILLS.

THE LAST LESSON WILL PROVIDE YOU WITH A BRIEF SUMMARY OF THE  
BASIC LANGUAGE .

go

THE FOLLOWING PROBLEMS WILL TEST YOUR LATEST SKILLS :

1. COMPUTE THE NET SALARY , RETIREMENT CONTRIBUTION  
AND TAX FOR N EMPLOYEES. NET PAY=GROSS SALARY-RETIREMENT-TAX  
USE SUBROUTINES TO CALCULATE :

#### A) RETIREMENT CONTRIBUTION

IF SALARY <\$800 , R=\$0.0

IF SALARY <\$2500 , R=\$10.

IF SALARY >\$2500 , R=\$20,

#### B) TAX

IF SALARY <\$600 , T=\$5 ; OTHERWISE  
T=.03(SALARY)

THE DATA IS:  
EMPLOYEE

GROSS SALARY



|       |        |
|-------|--------|
| JONES | 3000   |
| SMITH | 5500   |
| BROWN | 475    |
| DALEY | 10,500 |
| BERRY | 4300   |

2. WRITE A RECURSIVE PROGRAM TO SUM THE NUMBERS  
FROM X TO Y . READ IN YOUR OWN DATA AND TEST YOUR PROGRAM .

go

IF YOU WANT TO EXECUTE THESE PROGRAMS NOW , THEN  
REPLY : YES , OTHERWISE NO

no

DO YOU WANT TO TERMINATE YOUR INSTRUCTION SESSION ?  
(REPLY : YES OR NO )

no

IF YOU WANT TO EXECUTE PROGRAMS AT THIS TIME  
THEN REPLY : YES ; OTHERWISE REPLY : NO AND THE INSTRUCTION  
WILL CONTINUE .

no

REPLY WITH THE NUMBER OF THE LESSON YOU WISH TO COVER



EXECUTION BEGINS...

\*\*\* LESSON 7 \*\*\*

THIS LESSON SUMMARIZES THE BASIC DEFINITIONS AND BASIC STATEMENTS THAT YOU HAVE LEARNED IN CAI-BASIC .

A. BASIC DEFINITIONS

1. ALPHA-NUMERIC CHARACTERS :

A) DIGITS : 0-->9

B) LETTERS : A-->Z

C) SPECIAL CHARACTERS : \* \* / + - ( ) = ' , \$

2. STATEMENT NUMBERS : ONE TO FOUR DIGITS (0-->9999)

3. STRING : ANY SEQUENCE OF ALPHA-NUMERIC CHARACTERS ENCLOSED IN 'SINGLE' QUOTES . EG. 'HELP'

GO

4. NUMBERS : (LIMITED TO 9 DIGITS)

A) INTEGERS : DIGITS WITH NO FRACTIONAL PART . EG. 5,7,10

B) REAL : DIGIT WITH A FRACTIONAL PART . EG. 5.0,7.31,-16.0  
A NUMBER MAY BE PRECEDED BY A SIGN (+,-) , BUT IS ASSUMED TO BE POSITIVE IF NONE IS GIVEN .

5. VARIABLES :

A) SIMPLE VARIABLES : A SINGLE LETTER , OR A SINGLE LETTER FOLLOWED BY A DIGIT . EG A,A1,B6,Z0

B) ALPHA VARIABLES : A SINGLE LETTER FOLLOWED BY A DOLLAR SIGN (\$) EG. A\$,V\$

C) SUBSCRIPTED VARIABLE : (SINGLE LETTER)

(1) SINGLE SUBSCRIPT : <<LETTER>> ( <<EXPRESSION>> )

(2) DOUBLE SUBSCRIPT : <<LETTER>> ( <<EXPRESSION,EXPRESSION>> )

EG





A(5),Z(5,10),X(A\*B,X\*\*2)

go

5. OPERATORS : (LISTED IN DECENDING HIERARCHY)

|                   |    |
|-------------------|----|
| A) EXPONENTIATION | ** |
| B) MULTIPLICATION | *  |
| C) DIVISION       | /  |
| D) ADDITION       | +  |
| E) SUBTRACTION    | -  |

6. EXPRESSIONS :

A) SINGLE NUMBER OR VARIABLE EG. 10,-.53,D,X(I,J)

B) BUILT IN FUNCTION EG. SQR(10),TAN(.75)

C) ARITHMETIC EXPRESSION : EXPRESSIONS SEPARATED BY OPERATORS AND GROUPED BY PARENTHESIS . EG. 5+6.0 , A\*\*2(3\*X-4)

go

7. RELATIONS

|       |        |                              |
|-------|--------|------------------------------|
| A) GT | A GT B | A GREATER THAN B             |
| B) GE | A GE B | A GREATER THAN OR EQUAL TO B |
| C) LT | A LT B | A LESS THAN B                |
| D) LE | A LE B | A LESS THAN OR EQUAL TO B    |
| E) NE | A NE B | A NOT EQUAL TO B             |
| F) =  | A = B  | A.EQUAL TO B                 |

B. BASIC STATEMENTS

1. REM << ANY SET OF ALPHA-NUMERIC COMMENTS >>

2. READ << VARIABLE,....,VARIABLE >>

3. END

go

4. LET << VARIABLE >> = << EXPRESSION >>

5. PRINT << 'STRING' OR EXPRESSION,....,'STRING' OR EXPRESSION >>  
OR SIMPLY PRINT

6. DATA << 'STRING' OR NUMBER,....,'STRING' OR NUMBER >>



```

7. RESTORE OR RESTORE$
8. IF << EXPRESSION >> << RELATION >> THEN << STATEMENT NUMBER >>
9. GO TO << STATEMENT NUMBER >>
10. ON << EXPRESSION >> GO TO << STATEMENT NUMBER,...,STATEMENT NUMBER >>
11. FOR << SIMPLE VARIABLE >> = << EXPRESSION >>
    TO << EXPRESSION >> STEP << EXPRESSION >>
    ....
    ...
    ...
NEXT << SIMPLE VARIABLE >>
12. DIM << LETTER >> ( << INTEGER EXPRESSION >> )          OR
    DIM << LETTER >> ( << INTEGER EXPRESSION,INTEGER EXPRESSION >> )
13. GOSUB << STATEMENT NUMBER >>
    ...
    ...
    ...
RETURN

```

go

THAT CONCLUDES YOUR INSTRUCTION WITH CAI-BASIC .  
HOWEVER , YOU ARE INVITED TO USE WHATEVER FACILITIES OF CAI-BASIC  
YOU DESIRE AT ANY TIME . IF YOU HAVE NOT RUN PROGRAMS UNDER  
THE OS/BATCH MODE (PUNCHING YOUR OWN CARDS AND HANDING THEM  
ACROSS THE COUNTER TO BE RUN), YOU SHOULD GET THE BASIC MANUAL,  
TECHNICAL NOTE NR. 0211-12 , IN ROOM I-147 TO FIND THE PROPER  
JOB CONTROL CARDS REQUIRED . GOOD LUCK WITH THE COMPUTER , AND  
REMEMBER IT ONLY DOES WHAT YOU TELL IT TO DO - GIGO (GARBAGE IN  
GARBAGE OUT ).

DO YU WANT TO TERMINATE YOUR INSTRUCTION SESSION ?  
(REPLY : YES OR NO )

no

IF YOU WANT TO EXECUTE PROGRAMS AT THIS TIME  
THEN REPLY : YES ; OTHERWISE REPLY : NO AND THE INSTRUCTION  
WILL CONTINUE .

yes

\*\*\*\*\* CAIBASIC COMPILER \*\*\*\*\*

THE CAIBASIC COMPILER IS A LINE BY LINE INTERPRETER .THE BASIC



INTERPRETER ACCEPTS STANDARD BASIC STATEMENTS , AND IT ANALYZES EACH BASIC STATEMENT AS IT IS INPUT . AN ADDED FEATURE OF THE CAIBASIC COMPILER IS AN EDITING AND A DEBUGGING ROUTINE THAT ALLOWS THE USER TO ADD , DELETE AND CORRECT STATEMENTS ; AND TO GET A LISTING OF ALPHA-NUMERIC AND NUMERIC DATA USED , AND A TRACE OF ALL SIMPLE VARIABLES AS THEY ARE ASSIGNED VALUES IN THE PROGRAM.

THE DEBUG FEATURE IS USED BY ADDING THE KEY WORD DEBUG AS A STATEMENT TO YOUR PROGRAM .

THE EDITING MODE IS AVAILABLE TO THE USER WHEN AN EXECUTION ERROR OCCURS AND AFTER SUCCESSFUL EXECUTION .

IN THE EVENT OF AN INPUT ERROR THE INTERPRETER WILL ANALYZE THE ERROR AND PRINT AN ERROR MESSAGE. IF THIS OCCURS FIND THE ERROR ,AND INPUT THE CORRECT BASIC STATEMENT FOR THE CURRENT LINE OF INPUT .

(NOTE : TYPING ERRORS CAN BE DELETED BY TYPING FOUR DOLLAR SIGNS (\$\$\$\$) ON THE SAME LINE AS THE ERROR , HITTING CARRIAGE RETURN , AND INPUTTING THE LINE AGAIN .)

\*\*\*\* CAIBASIC EXECUTION \*\*\*\*

INPUT BASIC PROGRAM NOW (ONE LINE AT A TIME )

```
let x=1
gosub 100
print x
go to 9999
  100 let x=x+1
    if x gt 3 then 150
    gosub 100
  150 return
9999 end

4
```

DO YOU WANT TO EDIT YOUR PROGRAM  
REPLY : YES OR NO .

yes



\*\*\*\* CAIBASIC EDIT MODE \*\*\*\*

BY USING THE REFERENCE NUMBERS LISTED TO THE LEFT OF YOUR BASIC PROGRAM STATEMENTS YOU MAY ADD , DELETE , OR CORRECT ONE LINE OF THE PROGRAM AT A TIME . IN ALL EDITING THE FIRST STEP IS TO INPUT THE PROGRAM STATEMENT REFERENCE NUMBER AND HIT THE CARRIAGE RETURN . THE SECOND STEP DEPENDS ON WHAT EDITING YOU DO :

1. DELETE

THE BASIC STATEMENT REFERENCED IS DELETED BY TYPING THE LETTERS DEL

2. CORRECT

TO CORRECT THE BASIC STATEMENT REFERENCED TYPE IN THE COMPLETE CORRECT BASIC STATEMENT .

3. ADD

A BASIC STATEMENT IS ADDED 'AFTER' THE BASIC STATEMENT REFERENCED BY TYPING THE LETTERS ADD1 FOLLOWED BY THE BASIC STATEMENT . ALL BLANKS FOLLOWING THE LETTERS ADD1 WILL BE INCLUDED IN THE BASIC STATEMENT . TO PLACE A STATEMENT 'BEFORE' THE FIRST STATEMENT IN THE PROGRAM , USE THE REFERENCE NUMBER 0 .

```

1      LET X=1
2      GOSUB 100
3      PRINT X
4      GO TO 9999
5          100 LET X=X+1
6              IF X GT 3 THEN 150
7                  GOSUB 100
8          150 RETURN
9      9999 END
    
```

INPUT REFERENCE NUMBER NOW  
8

INPUT EDITING NOW (DEL,BASIC STATEMENT,ADD1....)

150 let x=x+1

MORE CORRECTIONS TO BE MADE ?? REPLY: YES ;  
OR REPLY: NO AND THE EDITED PROGRAM WILL BE EXECUTED .

yes





```

1      LET X=1
2      GOSUB 100
3      PRINT X
4      GO TO 9999
5          100 LET X=X+1
6              IF X GT 3 THEN 150
7                  GOSUB 100
8          150 LET X=X+1
9      9999 END

```

INPUT REFERENCE NUMBER NOW  
8

INPUT EDITING NOW (DEL,BASIC STATEMENT,ADD1....)

add1 return

MORE CORRECTIONS TO BE MADE ?? REPLY: YES ;  
OR REPLY: NO AND THE EDITED PROGRAM WILL BE EXECUTED .

no

```

LET X=1
GOSUB 100
PRINT X
GO TO 9999
    100 LET X=X+1
        IF X GT 3 THEN 150
            GOSUB 100
    150 LET X=X+1
        RETURN
9999 END

```

7



DO YOU WANT TO EXECUTE ANOTHER PROGRAM  
REPLY : YES OR NO .

no

DO YOU WANT TO TERMINATE YOUR INSTRUCTION SESSION ?  
(REPLY : YES OR NO )

yes

R; T=6.99/23.96 15.56.04



## BIBLIOGRAPHY

1. FENICHEL, R. R. , WEIZENBAUM, J. , and YOCHELSON, J.C. , "Program to Teach Programming," COMMUNICATIONS OF THE ACM, v.13, March 1971.
2. FENICHEL, R. R. , "The Teach System" (unpublished work, PROJECT MAC M-388, MIT, Cambridge, Mass. , 1969).
3. HARE, V. C. , BASIC PROGRAMMING, Harcourt, Brace, and World, Inc., 1970.
4. KERR, E. G. , TING, E. G. , and WALDEN, W. E. , "A Control Program for Computer Assisted Instruction," 24TH CONFERENCE PROCEEDINGS OF THE ACM, 1969.
5. NOLAN, R. L. , INTRODUCTION TO COMPUTING THROUGH THE BASIC LANGUAGE, Holt, Rinehart, and Winston, Inc. , 1969.
6. OETTENDER, A. G. , RUN, COMPUTER, RUN, Harvard University Press, 1969.
7. SMITH, W. R. , and YOUNG, J. L. , "Computer Assisted Instruction," MANAGEMENT JOURNAL, Naval Postgraduate School, March 1971.
8. STOLUROW, L. M. , "Computer Assisted Instruction," REPORT OF A CONFERENCE, U. S. Department of HEW, Washington, 1969.
9. U. S. COMMISSION ON INSTRUCTIONAL TECHNOLOGY, "To Improve Learning," U. S. Government Printing Office, Washington, 1970.
10. U. S. NAVAL POSTGRADUATE SCHOOL TECHNICAL NOTE NR. 0211-12 "Basic Language Manual," April 1971.



# INITIAL DISTRIBUTION LIST

|                                                                                                            | No. Copies |
|------------------------------------------------------------------------------------------------------------|------------|
| 1. Defense Documentation Center<br>Cameron Station<br>Alexandria, Virginia 22314                           | 2          |
| 2. Library, Code 0212<br>Naval Postgraduate School<br>Monterey, California 93940                           | 2          |
| 3. ENS Alan B. Roberts, Code 53Ro<br>Naval Postgraduate School<br>Monterey, California 93940               | 1          |
| 4. LT Thomas A. Barry, USN<br>2730 Murray Ave.<br>Norfolk, VA 23518                                        | 1          |
| 5. Assoc Professor Robert C. Bolles, Code 53 Bq<br>Naval Postgraduate School<br>Monterey, California 93940 | 1          |





(Security classification of title, body of abstract and indexing annotation must be entered when the overall report is classified)

|                                                                  |                                                                             |                                                         |  |
|------------------------------------------------------------------|-----------------------------------------------------------------------------|---------------------------------------------------------|--|
| ORIGINATING ACTIVITY (Corporate author)                          |                                                                             | 2a. REPORT SECURITY CLASSIFICATION                      |  |
| Naval Postgraduate School<br>Monterey, California 93940          |                                                                             | Unclassified                                            |  |
| REPORT TITLE                                                     |                                                                             | 2b. GROUP                                               |  |
| CAI-BASIC<br>A program to Teach the Programming Language "BASIC" |                                                                             |                                                         |  |
| 3. DESCRIPTIVE NOTES (Type of report and, inclusive dates)       |                                                                             |                                                         |  |
| Master's Thesis; September 1971                                  |                                                                             |                                                         |  |
| 4. AUTHOR(S) (First name, middle initial, last name)             |                                                                             |                                                         |  |
| Thomas Anthony Barry                                             |                                                                             |                                                         |  |
| REPORT DATE                                                      | 7a. TOTAL NO. OF PAGES                                                      | 7b. NO. OF REFS                                         |  |
| September 1971                                                   |                                                                             | 10                                                      |  |
| 5a. CONTRACT OR GRANT NO.                                        | 9a. ORIGINATOR'S REPORT NUMBER(S)                                           |                                                         |  |
| 5b. PROJECT NO.                                                  |                                                                             |                                                         |  |
| 5c.                                                              | 9b. OTHER REPORT NO(S) (Any other numbers that may be assigned this report) |                                                         |  |
| 5d.                                                              |                                                                             |                                                         |  |
| 10. DISTRIBUTION STATEMENT                                       |                                                                             |                                                         |  |
| Approved for public release; distribution unlimited.             |                                                                             |                                                         |  |
| 11. SUPPLEMENTARY NOTES                                          |                                                                             | 12. SPONSORING MILITARY ACTIVITY                        |  |
|                                                                  |                                                                             | Naval Postgraduate School<br>Monterey, California 93940 |  |
| 13. ABSTRACT                                                     |                                                                             |                                                         |  |

This paper presents a computer aided instruction program that fulfills the objectives of teaching a simple programming language, interpreting student responses, and executing and editing student programs. The CAI-BASIC program is written in FORTRAN IV, level G, and executes on IBM-2741 terminals while running under the CP-67/CMS time sharing system on the U. S. Naval Postgraduate School's IBM-360/67 computer system. The instructional phase of CAI-BASIC presents the fundamentals of "BASIC," a simple user oriented language, in seven lessons. During the instructional sessions the student is presented material and, based on his response to questions, he is routed to the next sequence of instructions. The execution phase of CAI-BASIC allows execution of "BASIC" programs, and has an optional debug feature that provides a trace of program variables to aid the student in finding programming errors. In the event of programming errors the user may enter an edit mode to correct mistakes in his program.



KEY WORDS

LINK A

LINK B

LINK C

| NAME        | ROLE   |
|-------------|--------|
| 1. [Name]   | [Role] |
| 2. [Name]   | [Role] |
| 3. [Name]   | [Role] |
| 4. [Name]   | [Role] |
| 5. [Name]   | [Role] |
| 6. [Name]   | [Role] |
| 7. [Name]   | [Role] |
| 8. [Name]   | [Role] |
| 9. [Name]   | [Role] |
| 10. [Name]  | [Role] |
| 11. [Name]  | [Role] |
| 12. [Name]  | [Role] |
| 13. [Name]  | [Role] |
| 14. [Name]  | [Role] |
| 15. [Name]  | [Role] |
| 16. [Name]  | [Role] |
| 17. [Name]  | [Role] |
| 18. [Name]  | [Role] |
| 19. [Name]  | [Role] |
| 20. [Name]  | [Role] |
| 21. [Name]  | [Role] |
| 22. [Name]  | [Role] |
| 23. [Name]  | [Role] |
| 24. [Name]  | [Role] |
| 25. [Name]  | [Role] |
| 26. [Name]  | [Role] |
| 27. [Name]  | [Role] |
| 28. [Name]  | [Role] |
| 29. [Name]  | [Role] |
| 30. [Name]  | [Role] |
| 31. [Name]  | [Role] |
| 32. [Name]  | [Role] |
| 33. [Name]  | [Role] |
| 34. [Name]  | [Role] |
| 35. [Name]  | [Role] |
| 36. [Name]  | [Role] |
| 37. [Name]  | [Role] |
| 38. [Name]  | [Role] |
| 39. [Name]  | [Role] |
| 40. [Name]  | [Role] |
| 41. [Name]  | [Role] |
| 42. [Name]  | [Role] |
| 43. [Name]  | [Role] |
| 44. [Name]  | [Role] |
| 45. [Name]  | [Role] |
| 46. [Name]  | [Role] |
| 47. [Name]  | [Role] |
| 48. [Name]  | [Role] |
| 49. [Name]  | [Role] |
| 50. [Name]  | [Role] |
| 51. [Name]  | [Role] |
| 52. [Name]  | [Role] |
| 53. [Name]  | [Role] |
| 54. [Name]  | [Role] |
| 55. [Name]  | [Role] |
| 56. [Name]  | [Role] |
| 57. [Name]  | [Role] |
| 58. [Name]  | [Role] |
| 59. [Name]  | [Role] |
| 60. [Name]  | [Role] |
| 61. [Name]  | [Role] |
| 62. [Name]  | [Role] |
| 63. [Name]  | [Role] |
| 64. [Name]  | [Role] |
| 65. [Name]  | [Role] |
| 66. [Name]  | [Role] |
| 67. [Name]  | [Role] |
| 68. [Name]  | [Role] |
| 69. [Name]  | [Role] |
| 70. [Name]  | [Role] |
| 71. [Name]  | [Role] |
| 72. [Name]  | [Role] |
| 73. [Name]  | [Role] |
| 74. [Name]  | [Role] |
| 75. [Name]  | [Role] |
| 76. [Name]  | [Role] |
| 77. [Name]  | [Role] |
| 78. [Name]  | [Role] |
| 79. [Name]  | [Role] |
| 80. [Name]  | [Role] |
| 81. [Name]  | [Role] |
| 82. [Name]  | [Role] |
| 83. [Name]  | [Role] |
| 84. [Name]  | [Role] |
| 85. [Name]  | [Role] |
| 86. [Name]  | [Role] |
| 87. [Name]  | [Role] |
| 88. [Name]  | [Role] |
| 89. [Name]  | [Role] |
| 90. [Name]  | [Role] |
| 91. [Name]  | [Role] |
| 92. [Name]  | [Role] |
| 93. [Name]  | [Role] |
| 94. [Name]  | [Role] |
| 95. [Name]  | [Role] |
| 96. [Name]  | [Role] |
| 97. [Name]  | [Role] |
| 98. [Name]  | [Role] |
| 99. [Name]  | [Role] |
| 100. [Name] | [Role] |

WT

|     | NAME                | ROLE               |
|-----|---------------------|--------------------|
| 1.  | Mr. J. Edgar Hoover | Director           |
| 2.  | Mr. Clegg           | Chief Clerk        |
| 3.  | Mr. Glavin          | Assistant Director |
| 4.  | Mr. Ladd            | Assistant Director |
| 5.  | Mr. Nichols         | Assistant Director |
| 6.  | Mr. Rosen           | Assistant Director |
| 7.  | Mr. Tracy           | Assistant Director |
| 8.  | Mr. Egan            | Assistant Director |
| 9.  | Mr. Gurnea          | Assistant Director |
| 10. | Mr. Harbo           | Assistant Director |
| 11. | Mr. Hendon          | Assistant Director |
| 12. | Mr. Pennington      | Assistant Director |
| 13. | Mr. Quinn           | Assistant Director |
| 14. | Mr. Nease           | Assistant Director |
| 15. | Mr. Tamm            | Assistant Director |
| 16. | Mr. Winterrowd      | Assistant Director |
| 17. | Mr. Mohr            | Assistant Director |
| 18. | Mr. Casper          | Assistant Director |
| 19. | Mr. Callahan        | Assistant Director |
| 20. | Mr. Connelley       | Assistant Director |
| 21. | Mr. Felt            | Assistant Director |
| 22. | Mr. Gale            | Assistant Director |
| 23. | Mr. Rosen           | Assistant Director |
| 24. | Mr. Sullivan        | Assistant Director |
| 25. | Mr. Tavel           | Assistant Director |
| 26. | Mr. Trotter         | Assistant Director |
| 27. | Mr. Tele. Room      | Telephone Operator |
| 28. | Miss Gandy          | Secretary          |

|    |
|----|
| WT |
|----|

|      |  |
|------|--|
| ROLE |  |
|------|--|

WT

### Computer aided instruction

Computer assisted instruction

Programmed instruction













130779  
22 FEB 78

22477  
25507

Thesis  
B24222  
c.1

Barry

CAI-BASIC; a program  
to teach the program-  
ming language "BASIC".

130779

22 FEB 78

22477  
25507

Thesis  
B24222  
c.1

Barry

CAI-BASIC; a program  
to teach the program-  
ming language "BASIC".

130779

thesB24222

CAI-BASIC :



3 2768 002 01476 3

DUDLEY KNOX LIBRARY